MASTER IN COMPUTER SCIENCE PROFESSIONAL FOCUS IN DATA SCIENCE

Using Prior Information to ImproveCrop/Weed Classification by MAV Swarms a review and reproduction

Hubermont, Antoine

Award date:
2021

Awarding institution:
University of Namur

Université de Namur
Faculté d'informatique
Année académique 2020–2021

Using Prior Information to Improve Crop/Weed Classification by MAV Swarms: a review and reproduction.

Antoine Hubermont

Maître de stage : V. Trianni

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Pr. E. Tuci

Co-promoteur : G. Maitre

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Acknowledgements

First of all, I would like to thank Pr. Elio Tuci and Guillaume Maître for their valuable advices and their support. I would also like to thank Vito Trianni (ISTC-CNR) for giving me the opportunity to work on this exciting topic and for his counseling.

Secondly, I would like to thank Federico Magistri for his precious advice on the Faster RCNN and for sharing the first dataset. I also want to thank Carlos Carbone for introducing me to the data generation with Unity3D and having shared the second dataset.

I want to thank the Faculty of Computer Science (University of Namur) for having helped me evolve in a fulfilling environment that made me progress toward the artificial intelligence field.

Finally, I would like to thank my friends and family who encouraged me in my goals and for their unfailing support during all the development of this work.

Abstract:

This master thesis proposes an introduction to Region-Based CNN with the use of Faster RCNNs via the reproduction of a scientific protocol [24] in the context of precision agriculture. Two Faster RCNNs embedded on swarm MAV have been developed to detect weeds in sugar beet crops simulated on Unity3D. These Faster RCNNs are of different depths. The objective is to increase the performance of the shallowest Faster RCNN to optimize computational resource consumption. The detections from previous passes of a MAV over the same area will be used to improve the detections of this Faster RCNN. The performances are compared. The past detections have no impact on the detection performances of the shallowest Faster RCNN. A confusion is made between epochs and iterations during the development and the training of the Faster RCNNs. The TorchVision library is used in first time. Then it is replaced by the maskrcnn-benchmark framework. The dataset used has a lack of images. As a result, it was not easy to train these networks. The results obtained in the framework and those obtained in the article defining the methodology [24] will be compared. Finally, this work will propose different ways of improvements.

Key-words: Artificial intelligence, Computer Vision, Object detection, Faster RCNN, MAV, Swarm, Precision agriculture, Weed control.

Résumé:

Ce mémoire propose une introduction aux Region Based CNNs avec l'utilisation de Faster RCNNs via la reproduction d'un protocole scientifique [24] dans le cadre de l'agriculture de précision. Deux Faster RCNNs embarqués sur des MAVs en essaim ont été développés pour détecter les mauvaises herbes dans les cultures de betteraves sucrières simulées sur Unity3D. Ces Faster RCNNs sont de profondeur différente. L'objectif est d'augmenter les performances du Faster RCNN le moins profond pour optimiser la consommation de ressource de calcul. Les détections issues des passages précédents d'un MAV au-dessus d'une même zone seront utilisées pour améliorer les détections de ce Faster RCNN. Les performances sont comparées et il en résulte que les détections passées n'ont pas d'impact sur les performances de détection du Faster RCNN le moins profond. Pour développer et entraîner les Faster RCNN, une confusion a été faite entre epochs et itérations. La bibliothèque TorchVision a d'abord été utilisée pour laisser place au framework maskrcnn-benchmark. Ensuite, le dataset utilisé manquait d'images et il n'a pas été simple d'entraîner ces réseaux. Les résultats obtenus dans le cadre de travail et ceux obtenus dans l'article définissant la méthodologie [24] seront comparés. Finalement, différentes pistes d'améliorations de ce travail seront proposées.

Mot-clefs: Intelligence artificielle, Vision par ordinateur, Détection d'objets, Faster RCNN, VAMM, Agriculture de précision, contrôle des mauvaises herbes.

Contents

Chapter 1

Introduction

Since the beginning of humanity, finding food is a crucial activity. Following the technological advances, agricultural practices evolved to become more efficient across the ages. The weeds are a real threat to the entire crop. They reduce crop yield, attract parasites, and require a lengthy and costly human operation to remove them. Commonly, this problem is solved with pesticides to prevent the weeds and parasites in the crops. This approach has three significant problems: damaging the biodiversity, high cost, and parasites or weeds can develop resistance to the chemicals. Today, the protection of biodiversity is a core question in the public debates and the role of agriculture is major. Therefore, develop sustainable agriculture is one of the big projects of our decade. Also, the use of chemicals in agriculture is one of the causes of biodiversity degradation. The weeds in crops are a significant reason for the intensive usage of chemicals.

The solution of this problem can be reduced by finding the solution of three sub-problems: Weed recognition, Weed removal and scalability. IT technologies are suitable to handle these problems. Moreover, the domain of smart agriculture becomes more and more critical and is an emerging solution to develop sustainable agriculture. Among these technologies, artificial intelligence and robotics constitute a great tool combination to remove the weeds in crops without the help of chemicals or via a costly and long human process.

The article *Using prior information to improve crop/weed classification by MAV Swarms* [24] proposes a solution in this direction. The authors develop an approach with the help of swarm robotics, Micro Aerial Vehicle (MAV) and machine learning. This approach consists of deploying MAV swarms across fields to detect the weeds in sugar beet crops. Computer vision techniques coupled to deep learning neural networks provide autonomy to the MAV to perform their detection tasks. This solution covers the weed recognition, the solution scalability and also brings a solid basis to develop the weed removal part. Regarding the scalability part, MAVs are well-designed to cover a large area and the swarm architecture brings a powerful communication and interaction system between the MAVs. A convolutional network is embedded in the MAV to perform the weed recognition and a probability map of the previous detection is also added to improve the detection speed. For the weed removal part, multiples solutions are possible. For example, communication with autonomous machines on the

grounds could be helpful. Also, articulating arms on the MAV can be developed in further works.

In a first time, the aim of this work is to reproduce and discuss the approach illustrated in [24]. Also, the methodology provides a clear understanding of the Region based convolutional network. First of all, a first section explains and defines the subject and its related techniques with a state-of-the-art review.

Secondly, a complete reproduction of the study is proposed. In this section, two Faster RCNNs will be developed to achieve the weeds detection on crop images. To perform object detection tasks, these models will be constructed on two different convolutional neural networks. A first discussion on their results with RGB images will be discussed. In a second phase, a probability map will be added to the input images to try to improve the detection score of these models.

Finally, a discussion is proposed to deepen the development and propose improvements in further works.

Chapter 2

State of the art

2.1 Precision agriculture

The agricultural sector is crucial for our societies. Due to its importance, it is at the center of many discussions and debates. Agriculture is also a crucial sector to fight against hunger in the world with the optimization of the yield. Also, agriculture is a crucial point to fight against global warming due to its energy and resource consumption. This industry consumes many rare and precious resources such as water and polluting with its consumption in pesticides and petrol. The economic situation of the different actors of agriculture is also an essential point because it remains the means of subsistence for a large number of people around the world. Because of its stakes, it is necessary to think of new ways to make it cleaner and more sustainable.To meet these challenges, Information Technologies (IT) are combined with the agricultural sector.

Precision agriculture is *the application of technologies and principles to manage spatial and temporal variability associated with all aspects of agricultural production for the purpose of improving crop performance and environmental quality* [34]. This definition was written in 1999 and gave an abstract sense of the concept: precision agriculture uses the available technologies to improve productivity and reduce resource consumption. In 2021, the precision agriculture benefits of a various range of efficient technologies: the connectivity between machines, sensors, satellites communications, data analysis... More precisely [6], some of the IT technologies used in precision agriculture are various like multimedia hardware (cameras...), Global Positioning System (GPS), sensors (in general), Unmanned Aerial Vehicles (UAV), robots... All these technologies give more possibilities to handle critical points in sustainable farm management like resource consumption.

These technologies allow a clear improvement in the efficiency of farms, such as weather forecasting, soil analysis for better yields or even disease detection. The contributions of precision agriculture improve the quality and productivity of agriculture. Also, it can help to reduce its consumption of energy, water and pesticides with better management of the activity. In addition, many tedious tasks previously reserved for humans are now performed by robots, which also

saves money in addition to preserving human life. This work takes advantage of these technologies to propose a solution to reduce pesticide consumption in fields with automatic detection of weeds in sugar beet fields. This work uses the swarm UAV connectivity to take pictures of the crops and make predictions based on the crop images with machine learning techniques. For a better understanding, the UAV family as well as the swarm architecture will be defined in the next section and will be followed by the machine learning section.

2.2 UAV Swarm architecture

In this section, the concept of UAV, Unmanned Aerial Vehicle, will be detailed. They cover large crop areas to take pictures with embedded materials like cameras and sensors. Also, these UAV are manipulated in a particular architecture called swarm.

2.2.1 Unmanned Aerial Vehicle family

A UAV is an aircraft system mainly used in dangerous situations or unreachable locations for a human. This type of aircraft embarks no human pilot. Instead, they can be controlled remotely with an operator or with AI systems to make them autonomous (with different levels of autonomy). As a result, they are used in various range of tasks in many sectors like military, civilian, police, agriculture... There are many different types of UAV, especially in their uses and in their sizes. For example, the *General Atomics MQ-9 Reaper* is a UAV used by the American army. This UAV has a 20m wingspan to fly at a high altitude (15000m), a payload of 1.7tons and can reach a maximum speed of 480km/h [2].
UAVs can also be used for more common tasks. For example, a french delivery enterprise, the DPD Group, uses a UAV to deliver packages to customers. This UAV is much more lightweight than the MQ-9 Reaper with a total weight of 4k [1]. Also, this UAV has 6 small electric rotors and can fly over a maximum of 15km at a speed of 30 km/h.

The smaller UAV are lightweight than the UAV. In the EU and for civil usage, they are fixed at a maximum of 25kg [25]. They generally measure between 50cm and 2m and have a structure like a plane (with wings) [3]. The Smaller UAV is small enough to be launched by a human. Another type of UAV exists and is smaller, the MAV (Micro Aerial Vehicle).

Micro aerial vehicle

The type of UAV used in this work are issued from the MAV sub-classes for their small sizes and their good maneuvering in narrow environment [26]. A MAV is designed to fly at low altitude (<330m) with a short flying time (<30min) due to its small size [44]. These characteristics are essential for their use in field areas and their swarm architectures. In order to go as close as possible to the target (plants), these drones have to be lightweight and small to navigate between crops without damaging the plants.

The lightweight proportion of this type of MAV implies a set of constraints, especially in terms of energy consumption, motorization, structure, weights and path planning strategy. Unlike bigger UAVs and their plane structure with wings, MAV can rely on one or more electrical rotors. The figure 2.1 shows a MAV with 4 rotors (quadcopter) mounted on a 50cm diameter platform. This MAV can carry 500gr and is able to take pictures from the sky and the ground with 2 onboard cameras [10].

Figure 2.1: A quadcopter MAV [10]

Due to their size, they are sensitive to wind and other forecast variations. Also, specific path planning strategies have to be developed to optimize the coverage of an area.

Field coverage strategy

In [4], a field coverage strategy is proposed with an improved random walker. A random walker strategy is a strategy inspired by the animals and others living entities in nature. It consists of an equivalent probability to move in a particular direction. For example, in the case of a random walker strategy in 2 dimensions, there are 4 possible directions: left, right, front and back.
The improved random walker covering strategy presented in [4] adds grids on the area. Each cell on the grid has to be cover at least one time. The goal is to cover all the cells in the zone. Each member of the swarm knows the already covered cells. Each member will randomly choose to visit the next uncovered cell with an equivalent probability to go in a particular direction.

In order to cover a large area, the energy autonomy has to be optimized and adapted to the MAV. In the same way, the weight of the equipment is a crucial element. Heavy batteries or heavy equipment cannot be embedded in these MAVs. Also, state-of-the-art detection frameworks can be very greedy in energy. They may require high-resolution pictures and also heavy computation power to achieve detection. One of the critical points of the work based on the article [24] is to limit this power consumption with a lightweight state-of-the-art detection algorithm and use multiple MAVs to take advantage of their numbers.

2.2.2 Swarms Robotics

Swarm robotics is a sub-branch of robotics that characterizes a set of robots as a swarm. This mode of organization between robots is directly inspired by nature and the behavior of some animals and insects. Indeed, the swarm population is composed of a large number of actors who coordinate themselves by communicating together to act with the same goal. A form of collective intelligence emerges from this swarm organization. Also, the decision-making is not centralized as in a classical robotics approach. Indeed, each individual has a minimal field of action and a low utility. Each individual has the same rights as the others and does not have a predefined role in the swarm. Also, there is no identification system inside the swarm and the individuals are not defined with ids. The strength of this organization is when these individuals are taken as a whole. They can achieve goals that can only be achieved as a group.

The swarm robotics is defined by three major characteristics [47]:

- Robustness.
  The swarm must be robust in its design. Because the swarm is composed of many single individuals, the swarm's mission is not threatened if one of them suffers a failure. Moreover, since the decision-making is not centralized, the swarm does not have a single point of failure. The other members of the swarm can continue to act autonomously without one of their members. The number of individuals also allows the swarm to maintain communication.

- Flexibility.
  The swarm must be flexible in the type of the task and its resolution. For example, the swarm may first be dispersed across an area to search and detect weeds. Then, the swarm regroups and coordinates its individuals to gather their strength and pull a weed. Flexibility is an essential element because it adds autonomy to the swarm.

- Scalability.
  A swarm must be scalable. The size of its population can vary and be composed of several size levels to be generalized. Concretely, if the swarm is functional with 5 individuals, it should also be functional with 50 individuals.

Therefore, a robot swarm is a robust, flexible and scalable form of organization. This organization can be handy in many use cases. Still, in [47] 4 major domains of application are defined.

- Cover an area.
  The flexibility of a swarm makes them effective in dispersing the swarm population across an area for inspection. In addition, dynamic allocation and communication within the swarm are helpful to perform adequate coverage. For example, a swarm of MAV can cover large areas of fields to detect weeds.

- Dangerous tasks
  A single individual in the swarm is simple and limited. Moreover, the

robustness characteristic of the swarm is helpful and the swarm can continue its mission even if some members are destroyed. In a dangerous zone, this configuration could circumvent the risk of global failure. Also, this prevents a human from performing this task. So, this risk could be transferred to a robot. For example, a rescue team sends a swarm robot in an area after an earthquake searching for survivors.

- Tasks with scalability over time. The swarm being a scalable entity, the size of its population can vary by several scales without disturbing its functioning. For example, a swarm of 10 elements is sent to extinguish a forest fire. If the fire spread, an operator could quickly send 10 additional individuals to the swarm to control the fire.

- Redundant tasks. A large number of individuals in the swarm is useful to send it to cover redundant tasks. Moreover, its robustness maintains this redundancy despite failures within it. For example, the swarm can be deployed as an emergency communication system where each individual ensures the redundancy of the communication nodes.

2.2.3 Conclusions

MAVs are ideal for use in swarm organization. Indeed, they are simple, agile and can effectively perform the task of covering a large area. The swarm configuration is suitable for taking pictures of the plants from the top and several positions. The robustness of the swarm, as well as the scalability, generalize this work to larger fields. Moreover, a MAV swarm is able to do multiple weed detections with the help of machine learning techniques. Also, MAVs can share information within the swarm.

2.3 Plant discrimination in precision agriculture

Plant detection and discrimination is a crucial point in precision agriculture. Plant detection algorithms recognizes and detects different types of plants in a field such as weeds for example. This detection increases the efficiency of weed control strategies by reducing the areas to be sprayed with chemicals and also detects plant diseases [28].

2.3.1 Approach

Several approaches are possible to detect plants as part of a weed control strategy. These approaches are fundamental because they determine the whole strategy. The strategy defines the hardware used and also the embedded equipment.

Ground approach

The ground approach consists of a plant by plant local treatment. An unmanned robot is commonly used for these types of tasks. For example, the autonomous farm tractor BONIROB [38] can drive between the crop rows and performs plant phenotyping and develop a map with all the gathered information. Plant phenotyping is the analysis of the observable characteristics of the plants. In

the case of sugar beets or weeds, these characteristics can be the color or the number of their leaves for examples. The BONIROB autonomous tractor is equipped, for a total weight of 4 tons, with multiple components such as 3D laser scanners, 3D Time of Flight cameras (produces a point cloud corresponding to the photographed scene)... All this equipment is dedicated to operating precise phenotyping of plants. This type of robot can also detect weed plants and thus reduce the spraying of insecticides with a precise local spraying [14] [39]. It is also evident that the ground approach can be achieved by humans. However, this method remains very expensive and tedious considering the surfaces to be covered.

The advantages of the ground approach are multiple. The robot can carry more material and more batteries for a more extended autonomy. Moreover, it can easily reach the plants and sprays precisely the weeds. However, this approach has limitations. The size and maneuverability of the robot must be controlled to avoid damaging the crops and soil. It is also necessary to take into account the weather conditions and ensure the robots are not stuck in the mud in the middle of a field. Moreover, these robots have not a global view of a field and cannot take advantage of the geometric patterns present in the crops.

Airborne approach

The airborne approach consists of the usage of images with a global view of an area. Multiple sensors and cameras are placed in altitude to do pictures or measures. In order to do this, UAVs or even satellite images are used to obtain a global view of an area. A high altitude point of view is required to take advantage of the geometric patterns in the fields. The geometric pattern is illustrated with a smaller scale in the images used in this work. For example, in Figure 3.3 the sugar beets are placed in parallel lines and form a row. These patterns are helpful because they are easily replicable in simulators to generate realistic images. They also improve the detection in machine learning algorithms. These algorithms use this type of pattern to detect objects in images. Machine learning algorithms will be detailed in the CNN architecture section. However, the use of UAVs is more appropriate in our case because the spatial resolution of UAVs (0.15m) is much more precise than multispectral satellites (from 1.64m to 4m) [23].

This view is declined in two categories. The first category is the high view and is characterized by images of less quality (more global but less precise). The second category is the low view with very high-quality images (less global but more precise). Useful information and maps can be extracted with the covering from a high view perspective. A quadcopter UAV takes pictures of the ground (30m from the ground) and then produces a map of weed infestation on a large area of a cornfield [32]. This solution determines the percentage of an area covered by weeds and provides an infestation level.

The limits of the airborne approach are the constraints in terms of weight and energy autonomy. Unlike the ground approach, the embedded material has to be optimized. Moreover, it is also necessary to monitor the weather conditions to ensure a safe flight of UAVs.

This work proposes a low view (3m from the ground) detection method to detect weeds. UAVs are too big for this altitude but MAVs are well designed for this altitude. The MAVs are used in swarm architecture to cover a large area without damaging the crops. This work also takes into account the airborne constraints and adapt the computing power required for the detection.

Conclusion

Both approaches have their advantages depending on the use and the objective. Therefore, it is interesting to combine these two techniques to optimize weed control. Typically, the airborne approach brings UAVs to assess the infestation of a field and define priority areas (a probability map of the infested area will be defined later). In parallel, the ground approach provides operational actions like weeds removal or spraying.

2.3.2 Image capture

Image capture is the technique used to obtain the images. An image is defined by the hardware used to do the caption. Each hardware or caption technique implies pros and cons depending on the usage.

Stereo vision

Stereo vision can combine the images of 2 cameras to obtain a 3D representation of the environment [20]. The stereo vision technique can be used with top view images and depth information to establish a skeleton of corn plants[17]. Each plant is then isolated and its core point is defined. The main disadvantage of stereo vision is its high sensitivity to light variations as well as the difficulty to recognize hidden or obscured elements in the image[9]. Their advantages are a low cost, images of high quality and the color information [45].

LIDAR

The principle of LIDAR (Light Detection And Ranging) is simple. The time taken by a light beam to return to its sender is recorded. Therefore, it is possible to compute the distance between the sender and a distant object with the speed of light. This process is applied for each point of the image to have a complete picture. The depth dimension in the image is added with the LIDAR technology. Ground robots can also embark this technology. In [45] 3D LIDAR sensors are embedded on ground robots to detect corn plants. A soil analysis is first performed and then the cloud of points collected between the soil and crop lines is segmented (cluster of lines).

RGB and infrared image

The biological characteristics of a plant can also be used, such as the reverberation of the chlorophyll [21]. A robot on the ground detects sugar beets and weeds. A camera recorded the ground illuminated by halogen spots as well as the infrared channel (NIR) to take advantage of the high reflection of the chlorophyll. It is also possible to use RGB images [22] and a NIR channel from a UAV [27] to take pictures of the ground. Images with only RGB channels

will be used to detect crops and weeds with the help of state-of-the-art machine learning techniques.

2.3.3 Machine learning and detection

Machine learning techniques can discriminate different elements in an image. In [22], an UAV discriminates beet plants and weeds using a RandomForest algorithm with images taken from an embedded camera.

The classification can also be performed using only the images taken by a UAV combined with machine learning algorithms [27] (detailed in the section next chapter). The RGB images as well as a NIR channel are used to train this type of algorithm. This algorithm can discriminate the soil crops and determine the precise location of a plant or a weed with pixel segmentation. Pixel segmentation is when each pixel in an image is classified (belongs to a plant, a weed or the background).

2.3.4 Constraints and limitations

There are several considerations for the detection process in the field. First of all, the process must not damage the plant or damage the equipment while doing the detection. For example, a large UAV could fly too close to the plants or a field robot could crush the crops.

Secondly, a high-quality detection is required to perform unique detection and localization for each plant (instance segmentation). There is also a need to have a high-quality image to have enough precision to perform accurate detections and segmentations of weeds. The detection hardware is also constrained by the weight of the material.

The maturity of the plants must also be taken into account. It is preferable to act as soon as possible on the weeds to limit their spreads, which will reduce the volume of pesticides sprayed. In this work, the plants have reached maturity and are detectable by UAVs [23].

2.3.5 Conclusions

For weed detection, this work uses MAVs organized in a swarm for their agility and for their very low altitude flight. Also, the ability of a MAV swarm to cover a large area at low altitude is used to take high-quality RGB images of a large area. A convolutional neural network is used to perform the classification and localization of plants and weeds on the image.

2.4 Region Based convolutional neural network

In this section, the machine learning techniques are detailed. The core concepts of the convolutional neural networks are developed. Then, a more complex convolutional neural network, the Region Based CNN, will be detailed. These techniques are used to extract features from the images taken by the MAV.

These features are then used to discriminate and localize weeds and plants on the image.

2.4.1 Convolutional Neural network

First of all, a neural network is a computational system inspired by the structural and functional properties of the central nervous system. A neural network can be trained with supervised learning algorithms. A learning algorithm is supervised when the input elements are linked with annotations. A neural network takes as input a set of raw elements and a set of annotated elements. In our case study, the annotations are the coordinates of the plants. The annotations also contain the associated plant class for each coordinate. A more detailed definition of these data is provided in the Dataset section.

The core concept of this study is to train a convolutional neural network to recognize and balance the extracted features from the input image. Training a network increases the probability from the network to correctly classify a new unseen element [13]. The network efficiency is the ability to generalize the predictions made on learning images on new images. The generalization of the predictions is a direct measure of the correct classification probability [19].

In a classification case, a vector containing the membership probability for each class is computed. In a regression case, a vector containing the approximate values for each element is computed. Then, the error between the predicted value and the annotation value is calculated (loss function). A coefficient based on this error is computed (Stochastic Gradient). Finally, the network propagates the coefficient using the backpropagation (a neuron weight update algorithm).

A convolutional neural network (CNN) is a neural network performing a convolution operation on the inputs at different stages. There are 4 distinct layers types in CNN [5] :

- Convolutional layer
  This layer applies filters (matrix) on images or feature maps (matrix). A dot product is operated between the filter and the images. These filters are applied over the images or feature maps by a sliding window. A sliding window has the same size as the filters and moves over the matrix by an iterative sliding movement to operate a complete covering.
  The filter has a smaller size than the other matrix, so the filter will move on the matrix by a sliding movement to cover it completely. The sliding of filters on an image or a feature map is called a sliding window.
  A dot product is computed between the filter and the area cover by the sliding window. The dot product is stored after each sliding movement of the window in a matrix called a feature map. In other words, the feature map contains the filtering results.

  In short, a convolutional layer applies a filter on an image and returns the results of the filtering in a feature map.

- Non Linearity layer
  This layer is an activation function. An activation function gives control

to a neuron output value. The most used activation function is the Rectify Linear Unit (ReLU). This function returns the input if it is positive and 0 otherwise.

- Pooling layer
  The pooling layer can lower the dimension of a matrix. The objective is to reduce the dimension of the feature maps to obtain more precision. The most commonly used reduction is the Max-Pool [5] which computed a lower dimension matrix containing the maximum values of the input matrix sub-regions.

- Fully connected layer
  In this layer, each neuron is connected to all the neurons of the previous layer. This layer is often used as the last layer to allow classification, where each neuron is a class-specific detector.

In Figure 2.2, a simple CNN is developed in the case of a classification task. In the first step, the input image is analyzed with multiples convolution filters to produce feature maps. This operation is repeated for each convolution layer on the feature maps computed on the previous layer. These phases are called feature extraction. The filters highlight the potential interesting elements to help the classification of the images. Then the last computed features maps are reduced with the pooling layer. Finally, the fully connected layers returned a membership probability for each class.

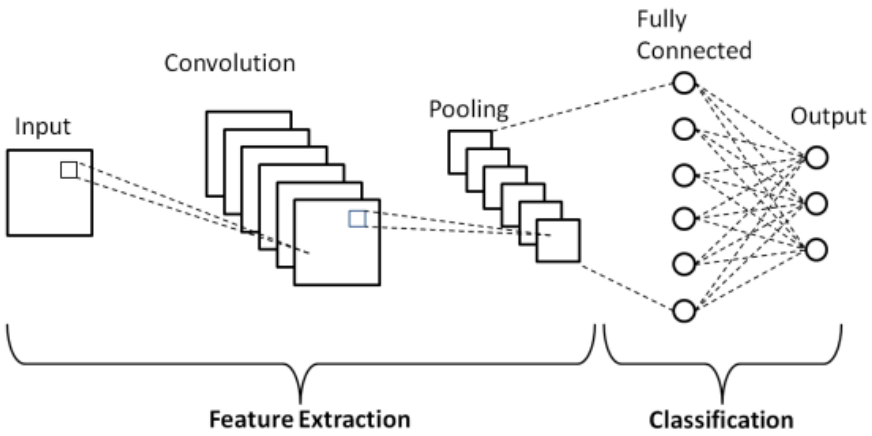

Figure 2.2: Abstract diagram of a convolutional neural network in a classification case [33]

2.4.2 Key elements

The development and the definition of the Region Based CNN is complex and involves several concepts. These terms and concepts are defined and explained in this dedicated section to treat them apart. A simple convolutional network is not able to classify different elements in a single image. Other convolutional network architectures can be used as feature extractors like the ResNet. A more complex convolutional network like the U-Net can classify pixels in images

(instance segmentation). However, CNNs like U-Net is not able to directly detect an object. The Region Based CNN are convolutional networks used in Computer vision to do object detection tasks. An object detection task requires two sub-tasks:

- Classify multiple elements in the image.
- Locate the detections in the image.

The strength of a Region based CNN is to reach the two sub-goals in a single architecture.

Region of interest

A region of interest (ROI) is an image portion. This portion has a high probability of containing an element of interest (an object). For example, an algorithm searches apples in a fruit basket image (Figure 2.3): an ROI might be a sub-regions of the image which looks like an apple. In this example, the ROI can contain areas of red pixels in a round shape.

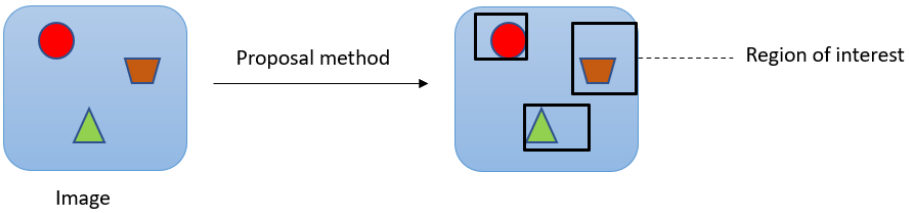

Figure 2.3: Example of a region of interest

A ROI is the result of a proposal method. A proposal method is an algorithm like the selective search algorithm [42] used for image recognition. More complex proposals methods have been developed like the region proposal network (RPN) used in Faster RCNN [35] where a CNN performs a binary classification between the background or an object. The RPN is detailed below in this section.

In the case of Region based CNN (RCNN and Fast RCNN), the selective search algorithm is used and the ROIs are extracted as follows:

1. Segmentation of the images in multiple sub-regions (based on filters).
2. Combination of similar regions in larger regions (iterative)

. The major problem of this algorithm is the large number of generated ROIs. More than 2000 ROIs per image are used in RCNN [12]). Each ROIs are then fed in a CNN, which requires a lot of computing power.

Bounding box

Object detection on an image can take the form of bounding boxes. A bounding box is a rectangle surrounding an object in an image (Figure 6.1. In the context of this work, the bounding box frames sugar beets and weeds. Unlike a simple object classification, the bounding box also locates the classified objects on the image.

The Region Based CNN uses bounding boxes to detect objects. To compute the boxes, the network needs a classifier and a regressor. The classifier assigns a belonging probability for each class to the bounding box depending on the framed area. A regressor is used to approximate the coordinates of the bounding box. The goal is to have an optimal overlap between the framed zone and the object on the image.

Intersection over Union

The Intersection over Union (IoU) metric measures the overlap between two elements. The IoU considers the height, width and location of two shapes and then compares their common area. For example, the IoU between a predicted bounding box and the ground-truth bounding box is widely used. In object detection tasks, IoU is used to differentiate false positives from true positives [36].

Fine-Tuning

The construction of a dataset is a core element for efficient learning. However, this construction can be very slow and costly dues to the difficulty of the dataset labeling and the data scarcity [30]. Also, training a network from scratch is a very long process, especially on a large dataset. A CNN already trained on a larger similar dataset can be helpful. These CNNs can be adapted to be used on specific data when these data are close to those used in the larger dataset.

Fine-Tuning is the process by which the weights of a trained CNN are loaded in a similar architecture CNN. This loading is done before the training. Also, the last fully connected layers have to be replaced to fit the new targets [41]. The other layers are frozen during the training (their weights will not be updated) to only train the last layers. This weight freezing keeps the general knowledge gained by the network and optimizes the last layer to the new data. This process is iterative. The number of frozen layers depends on the difference between the large dataset and the new data.

The Region Based CNNs presented in the following sections have been trained on larger datasets by their authors. For example, the ImageNet dataset [8], which contains millions of various class images (animals, flower, vehicle...) is often used to pretrained Region Based CNNs. However, the pretraining and the finetuning are illustrated here to understand the development of Region based CNN. These techniques will not be used in the methodology.

The pretrained CNNs used by the authors of the Region Based CNN are too heavy and require too much computing power to be embedded on a MAV. Lighter CNNs will be implemented and trained from scratch to solve this problem.

2.4.3 Region based CNN ensemble

The Region based CNN is the basis of various state-of-the-art models like the Faster RCNN [35] and the Mask RCNN [15] which will be detailed later. The

evolution of these models is developed here to fully understand the Faster RCNN used later in the development section. The architecture of this network and its approach are directly based on the Region based CNN family.

Region based CNN

The Region based CNN (RCNN) [12] proposes the concept of Region Of Interest (ROI) to perform object detection. These ROIs are provided to a CNN to compute much more accurate predictions. This type of supervised learning architecture is used for object detection on an image. To enable an efficient classification of the many ROIs, a score for each class is predicted. Then, the regions are compared between them. After the comparison, the region with the highest IoU is selected (non-maximum suppression). The objective is twofold, to minimize the classification error and then optimize the coordinates. The deviation between the approximated coordinates and the ground truth coordinates (annotation coordinates) has to be minimized. An ROI is classified as positive for training purposes if its overlap is ≥ 0.5 with the ground truth. This parameter is used to adjust the sensitivity of the detection.

The main characteristic of this CNN architecture is the use of all the proposals from an image. This process is an approach to the problem of locating elements in an image. Therefore, the CNN takes as inputs sub-parts of the images with a high probability of containing a feature (ROI). It is common in the literature to use weights from other networks trained on large image datasets and then finetuning the CNN to speed up the training [12]. The recommended training of this network is carried out using a CNN [18] with the weights from a network previously trained on the ImageNet dataset [8].

2.4.4 Fast RCNN

Fast RCNN [11] is an improved version of RCNN. The advantage of Fast RCNN is the extraction of the ROIs directly on the feature maps. These feature maps are the results of convolution phases. The selective search algorithm is also used to extract ROIs from them. This method dramatically reduces the number of inputs to the CNN: one image in Fast RCNN instead of thousands of ROIs for RCNN. According to the authors of [11], Fast RCNN increases the training speed of a CNN (VGG16) 9 times faster than the RCNN and also increases the accuracy of detections. ROIs can have variable sizes but the CNN classifier and regressor must have fixed and defined dimensional inputs. A new layer is introduced to handle this new method, the ROI pooling layer. This layer takes as input a feature map and the list of ROIs and returns this list of ROIs with their dimensions set to the same size. ROI pooling consists of two steps:

1. Subdivision of the ROI in several equal size sub-regions.
2. Maximum value is computed for each sub-regions and stored in the output matrix.

2.4.5 Faster RCNN

The Faster RCNN [35] is an improved version of the Fast RCNN and proposes a new method to obtain ROIs without the selective search (Figure 2.4). The Region Proposal Network (RPN) predicts ROIs and is more efficient. The Faster RCNN takes as input an image and performs several convolution phases. Then, the results of these convolutions, the feature maps, are fed into the RPN. The RPN generates anchors (defined below) and predicts the ROIs. Then, the feature maps obtained from the last convolution layer are concatenated with their ROIs given by the RPN. Finally, the Faster RCNN used these enhanced feature maps to predict the class and the location of the objects.

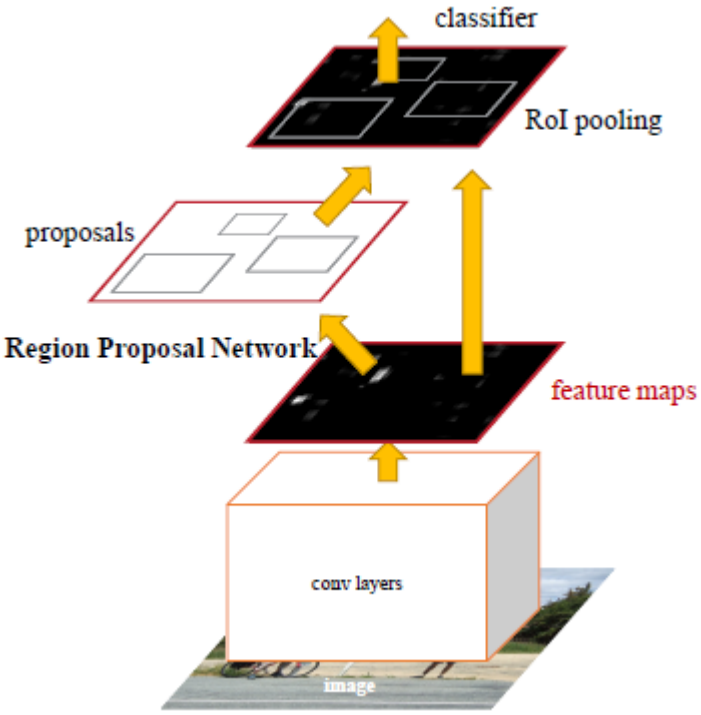

Figure 2.4: Faster RCNN architecture [35]

Anchors

The Anchor concept is introduced to perform analysis with multiple ratios and sizes inside a fixed position in the feature map to increase the precision. Anchors can be seen as camera lenses. Each anchor is like a specific camera lens to have a particular focus when the feature map is analyzed.

For each sliding window position in the feature map, rectangles are defined at the sliding window center. Each rectangle has a different scale and aspect ratio and is called an anchor. In the Faster RCNN there are by default 3 ratios (1:1,1:2,2:1) and 3 sizes, so 9 ($= k$) anchors are generated at each sliding window position [35]. For example, if the sizes of the anchors are

(50x50,100x100,200x200). All the possible anchors defined with a (2:1) ratio are rectangles of shape (100x50, 200x100, 400x200). Thanks to the multiple shape anchors, it is possible to have much more control and precision on the detection. Smaller anchors covered the smaller details and the larger ones are used to have a general overview of the feature map.

The anchors are used to arbitrarily cover elements in a feature map to help the ROI detections. When they are used in complex ROI proposal methods, the anchors will be classified as positive or negative depending on if they framed pixels belonging to a detectable object or not.

Region Proposal Network

The Region Proposal Network is a neural network feed with a feature map to return a list of predicted ROIs on the feature map (this process is also called region proposal). The RPN operates with the same approach as a sliding window. The input feature map is gridded and the RPN visits all the cells defined with the grid. At each cell, RPN draws multiple anchors defined on the center of the cell. Then, the RPN classifies each anchor depending on if this anchor frames an object or not. If the anchor frames an object, the RPN regresses the coordinates of the anchor. The goal is to fit at best the object shape. A positive predicted and regressed anchor is a ROI. The RPN does not make any differences between the class of the objects. The RPN only highlights the zones with the highest probability of containing an object in a feature map.

To train a RPN, the overlap (IoU score) of the zone covered by an anchor and the nearest ground truth object is calculated. An anchor is labeled as positive in its original version if the IoU is $\geq$ to 0.7 and as negative if the IoU is $\leq$ to 0.3. If the IoU is between these thresholds, the anchor is considered neither negative nor positive and will be ignored.

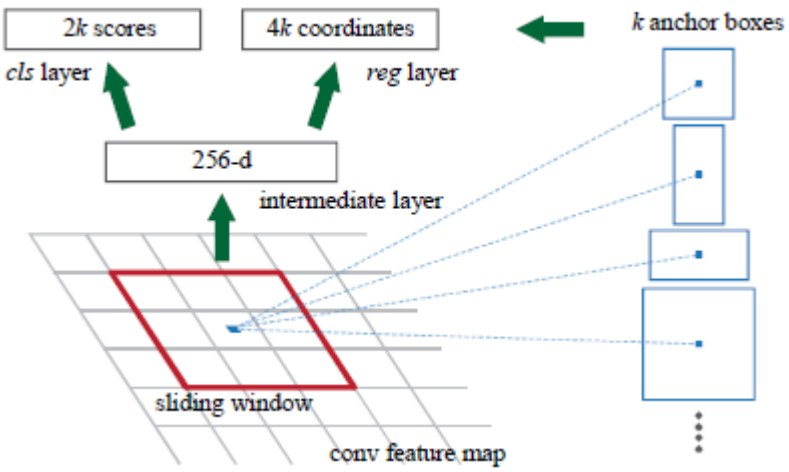

Figure 2.5: Region Proposal Network architecture [35]

In short, the action sequence (Figure 2.5) of a RPN per position is:

1. Sliding through the feature map and generate anchors (where k is the number of generated anchors at the position).
2. k numbers of ROIs are predicted on each position.
3. Classification of $2k$ classes: contains an object or not.
4. Regression of $4k$ values: refines coordinates of the bounding box.

For each anchor, the overlap of the zone covered by an anchor and the ground truth is calculated. An anchor is as positive if the framed zone is sufficiently interesting to be kept during the training of the RPN. The positive and negative threshold can be modified to vary the sensibility of the anchors. If the positive threshold is too high, only too few anchors will be label as positive. If the positive threshold is too low, too many anchors with few interests will be label as positive and will generate noise.

Training strategy

The particularity of the Faster RCNN is to contain another neural network (the RPN). Indeed, the RPN and the Faster RCNN use the same feature map to perform their respective classification and regression tasks. This implies sharing the convolution layers (the backbone) with the RPN and the Faster RCNN detector. In Figure 2.4, two arrows have their source on the feature map produced by the convolutional layers (the backbone). The left arrow point to the Region Proposal Network and the right arrow point to the RoI pooling. These arrows illustrate well the feature map sharing between the RPN and the Faster RCNN. This architecture often needs specific training because the weights have to be the same for both networks. The authors [35] recommend a training strategy in several phases.

1. RPN training (positive and negative anchors are generated in an image with an equal ratio).
2. A Faster RCNN detector (classification and regression) is trained separately with the anchors generated by the previous trained RPN.
3. RPN finetuning on the Faster RCNN frozen shared layers. Both networks share the same convolution layers.
4. Fine-tuning of the Faster RCNN detector (previously shared layers are defrosted).

However, this training strategy will not be applied in this work. This is because the lightweight CNN backbones used to produce the feature maps are much more shallow than those typically used. In this condition, training the Faster RCNN from scratch is possible. This method is therefore much more practical and faster and satisfies the shared convolution layer constraint.

Shared Convolutional layers

The RPN and Faster RCNN use the same feature maps and use the same convolution layer architecture. The CNN used to obtain the feature maps is called a backbone. The backbones are generally state-of-the-art CNNs like ResNet50 [16] or even VGG16 [40] but without the last fully convolutional layers to avoid the prediction steps. However, these networks require too much computational performance and are not suitable to be embedded on a MAV. To solve this problem, the authors [24] developed two lightweights Fully Convolutional Networks (FCN) backbones inspired by the ResNet architecture. An FCN is a convolutional network without a fully connected layer. Instead of these layers, specific convolutional layers are used with a fixed convolution filter dimension (1x1). An FCN with 5 convolution layers (FCN5) and another with 9 convolution layers (FCN9) are used as the backbone in this work and are defined in the following table :

FCN5	FCN9
7x7, 64, stride 2	
3x3, max pools, stride 2	
[3x3, 64] x 1	[3x3, 64] x 2
[3x3, 128] x 1	[3x3, 128] x 2
[3x3, 256] x 1	[3x3, 256] x 2
[3x3, 512] x 1	[3x3, 512] x 2

Table 2.1: Backbone architectures

The two first layers are common to both backbones. First, a convolution layer with 64 filters is applied to the input images with a 3x3 sliding window. The stride parameter defines the movement of a sliding window at each iteration over the image or the feature map. For example, a stride parameter set at 2 means the sliding window will shift 2 pixels at each iteration. After the convolution, a pooling layer is applied to the features maps. The max pools will only keep the maximum value of 3x3 subregions of the feature map.

For both backbones, a set of convolution layers are defined. The stride is defined at 1 and the filter numbers increasing from 64 to 512. Unlike the FCN5, the FCN9 uses a block of 2 convolution layers with the same filter number to have a deeper analysis of the image. The strategy is to capture high-level

features with a lower number of filters on bigger feature maps at the beginning. Then, more detailed features are extracted on smaller feature maps with a higher number of filters.
These FCNs will be used to extract accurate feature maps from the input images and the Faster RCNNs built on these networks will be compared in the results section.

2.5 Training

The training of a convolutional network such as the Faster RCNN is supervised. This means that the results of the detection are known and fixed in advance. Indeed, each training image is linked to its labeled information (ground truth). This labeled information contains all the information necessary to detect an object, such as the class of the object or its coordinates.

More generally, training a network consists in adapting the weights present in the neurons according to the task assigned to the network and the network performance to realize the task. This is done by minimizing a loss function measuring the efficiency of the network to perform this task.

2.5.1 Loss function

In the case of a Faster RCNN [35], two types of tasks are assigned to the network: a classification task and a regression task. Classification will determine if an object belongs to a specific class or not. The regression task predicts the coordinates of the bounding boxes and these coordinates have to be as close as possible to the ground truth bounding boxes coordinates. Each regressor regresses a vector of coordinates for a unique anchor size and ratio. The different regressors do not share their weights to match bounding boxes of varying sizes.

Here is the general Loss function of a Faster RCNN [35]:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

Therefore, there are two loss functions specific to each task. The general formula takes into account only the loss of the positive anchors and ignore the negative one. This is explained by multiplying p_i^* by the loss of the regression. Here is the detail of the loss function dedicated to the regression of the predicted coordinates of the bounding boxes:

$$L_{reg}(t_i, t_i^*) = Smooth_{L1}(t_i - t_i^*)$$

$$Smooth_{L1}(x,y) = \frac{1}{n}\sum_i z_i \quad where \quad z_i = \left\{ \begin{array}{cc} 0.5\,(x_i - y_i)^2/\,\beta & if \quad |x_i - y_i| < \beta \\ |x_i - y_i| - 0.5 * \beta & otherwise \end{array} \right\}$$

Regarding the classification loss function, it can be reduced to a binary classification loss by quantifying the margin of error of the network when it predicts the membership or non-membership of an object to a class:

$$L_{cls}(p_i, p_i^*) = -p_i^* log(p_i) - (1 - p_i^*) log(1 - p_i)$$

where [35]

- i is the anchor index inside a mini batch.
- p_i is the predicted probability for an anchor i to be positive (contain an object).
- p_i^* is the ground truth label of an anchor i.
  - $p_i^* = 1$ If the IoU between an anchor i and an ground truth bounding box is ≥ 0.7
  - $p_i^* = 0$ If the IoU between an anchor i and an ground truth bounding box is ≤ 0.3
- t_i is a vector containing the four predicted coordinates of the bounding box in the format: (abscissa, ordinate) of the center of the box, width and height.
- t_i^* is a vector containing the four coordinates of the box ground truth associated with the positive anchor.
- β F-score parameter here fixed at 1.
- N_{cls} is the size of the batch (number of images) used to train a Faster RCNN.
- N_{reg} is the number of anchor positions.
- λ is a parameter used by the authors of the Faster RCNN to balance the impact of loss linked to regression and was set to 10.

2.5.2 Training parameters

A set of parameters must be established to set up a training phase. The parameters are defined according to the network architecture type or the task that the convolutional neural network must perform. It should be noted that the setting of these parameters is variable and depends on many factors.

- Batch size: Corresponds to the number of images present in a batch. A batch is a group of elements generally smaller than the size of the training dataset and allows not to load the whole dataset in the network.
- Epoch: An epoch is performed when the entire training dataset has been loaded and processed by the network. A training of 100 epochs means that the dataset has been loaded 100 times in the network.
- Iteration: An iteration is done when an image batch has been loaded and processed by the network. A training of 100 iterations means that an image batch has been loaded 100 times in the network. The images present in the batch change at each iteration.

For example, a training dataset is composed of 1000 images. This dataset is split into 100 batches of 10 images each. A neural network trained on 100 iterations means that 100 batches have been loaded once in the network. At the end of the training, 1000 images are loaded to the network.
A neural network trained on 100 epochs means that the dataset has been loaded 100 times in the network. At the end of the training, 100000 images are loaded to the network.

2.5.3 Training optimiser

A Stochastic Gradient Descend algorithm (SGD) is used to minimize the loss function. The gradient is a mathematical measure evaluating how much the output of a function varies according to a modification of the input. The results of the loss function with all the possible values of the gradient can be seen as a parabola (Figure 2.6). This algorithm finds the best coefficient to minimize the loss function in searching the minimum point in this parabola. Once the best gradient is found, all the neuron weights are updated according to this gradient. There are two types of descent. The first type is the classic descent and is made after each epoch. The stochastic gradient descend is performed after each iteration.

This algorithm can be controlled with parameters like the learning rate. The learning rate defined the descent speed of the gradient along the curve (Figure 2.6). The learning rate has to be well balanced:

- Too high learning rate: the minimum point is missed.
- Too low learning rate: reaching the minimum point is very long to reach and there gradient risks to get stuck in local minimum (higher than the global minimum).

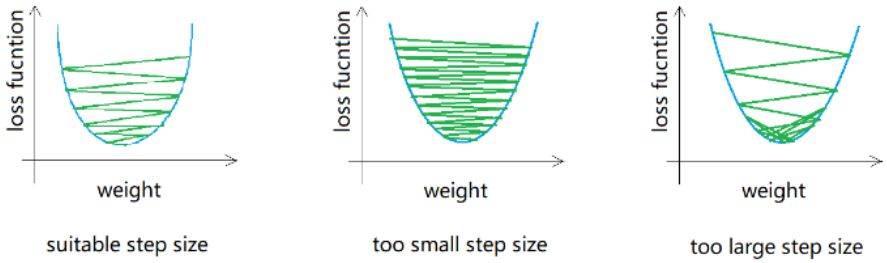

Figure 2.6: Effects on the learning rate in a SGD optimiser [7]

The SGD operates a gradient descent at each iteration. This method has one disadvantage. The gradient can have a wide range of values dues to the high frequencies of descents. This wide range can add noise and slower the training. The momentum parameter is introduced to smooth the value of the gradient in averaging the past computed gradient values.
The weight decay parameter is defined to prevent the weights from being updated too fast and reach an overfit of the network. The weight decay is a coefficient applied at each update on the weight to lower their value.

2.6 Evaluation

The evaluation measures the performance of the network detections on new images. Indeed, the performances evaluated on the images used during the training are not reliable. The network trained and updated its neuron weights to detect these images in particular. Its weights have been updated with the loss functions calculated on the basis of the training images. In a way, the network has learned by heart the evaluation images.
This bias artificially increases the performance of the network. Therefore, an evaluation of unseen images is required to have the best measure of the network performance.

2.6.1 Evaluation metrics

The performance of a network is evaluated according to objective metrics. When a network makes a prediction about the class membership of an element, several cases are possible. A positive prediction corresponds to membership in the class of this element. On the contrary, a negative prediction corresponds to a non-membership to the class.

- True positive (TP): An element is detected as positive by the network and this element is truly positive. The detected membership matches the ground-truth membership of the element.
- False positive (FP): An element is detected as positive by the network, which is negative. This element wrongly belongs to a class, the detected membership does not match the ground-truth membership of the element.
- False negative (FN): An element is detected as negative by the network, which is positive. This element wrongly not belongs to a class. The detected membership does not match the ground-truth membership of the element.

2.6.2 Precision

The precision (Pr) quantifies the rate of positive elements compared to the whole set of the predicted positive elements such as

$$Pr = TP \,/\, (TP + FP)$$

A low precision means the network wrongly classifies many elements. When a membership is predicted, the probability that the membership is correct is low. On the contrary, a high precision means the probability that the element is correctly classified is high.

However, precision is not enough to measure the overall performance of a network. Indeed, the accuracy only considers the reliability of the classification of an element only when this element is detected. Therefore, the precision score can be high even if the network forgets to detect many elements. For example,

if an image contains 10 elements of class X and 5 elements of class Y. In this image, a network has detected only one element of class X and two of class Y and classifies them correctly. For example, this network has a precision of 1 for the classes X and Y but has only detected 3 elements out of 15.

2.6.3 Recall

The recall (Rc) quantifies the rate of positive detected elements compared to all the positive elements present in the image such as

$$Rc = TP \,/\, (TP + FN)$$

. The recall measures the efficiency of the network to detect all the positive elements present in the image. In the above example, the network will have a recall for class Y of

$$2/(2+3) = 0.4$$

. The network correctly classified two y elements but did not detect the remaining 3 Y elements (assigned to the background class).

2.6.4 Fα-score

The F-score is a combination of precision and recall. Since each of these two measures taken separately is not effective in measuring the efficiency of a network, their combination is used to give an efficiency metric. This combination also increases the weight given to recall over precision with the real number α.

The general formula for the F-score is as follows:

$$F_\alpha = \frac{(1+\alpha^2) * (Pr * Rc)}{(\alpha^2 * Pr) + Rc}$$

In the context of this work, recall and precision have an equivalent weight, so the F-score formula becomes

$$F_1 = 2 * \frac{Pr * Rc}{Pr + Rc}$$

2.7 Prior information and probability map

In this work, MAVs are used in a swarm organization to perform weed detection on sugar beet crops. A MAV can cover large areas and often covers a zone previously covered by another swarm member. The intuition behind the prior information is to use detections previously made on a specific portion of a field to improve the detection performance. These previous detections are called prior information and are communicated to the swarm. In an area previously covered, a MAV can use the available prior information to improve the detection. Also, the MAV can correct the available information if he detects something different. In this work, the prior information is stored in a probability map.

A probability map [24] is a colored channel encoding the previous weed detections. A pixel with a black value (= 0) means this pixel is assigned to the background. The value of a colored pixel represents the probability of this pixel to be a weed. A high value means the pixel has a high probability of being a weed. This probability map is added to the input image and then sent to the Faster RCNN. This input image with the additional channels(RGB + probability map) will be sent to the network with a variable rate during the training. A probability map feeding rate is introduced during the training. The network has to learn how to classify when prior information is not available. This rate will be discussed in the results section. The probability map uses the results of the prediction made by the Faster RCNN.

The network predicts a confidence score(s) and two kinds of values for a detected object: a class(c) and the bounding box coordinates. These coordinates are the box center (x, y) and also the height(h) and width(w) of the box. The probability map for a detected object i) belonging the class c = Weed(W) is defined by the probability(P) of containing a weed at each point(x, y):

$$P(x,y) = \sum_{c_i=W} s_i \,.\, e^{-\left(\frac{(x-x_i)^2}{(2w_i^2} + \frac{(y-y_i)^2}{(2h_i^2}\right)}$$

. At this point, the probability map is only computed with the latest detection and does not consider the previous detections made by the networks for the same field portion. In Figure 2.7, a typical probability map is illustrated. This probability map is computed with the detections shown in Figure 3.7. The yellow spots represent the pixels with the highest probability to belonging to a weed. The pixels around the yellow spots have a smaller probability. Their colour tend towards the black the more the pixel is far from the center of the spots.

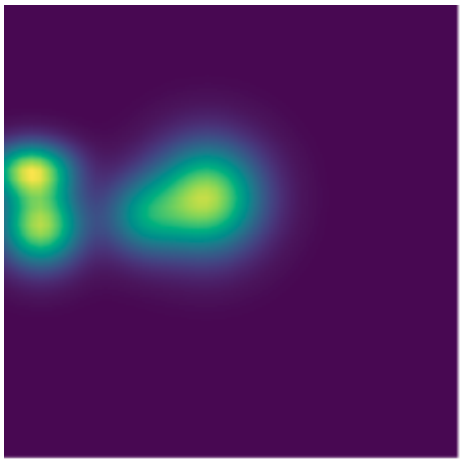

Figure 2.7: Probability map of the Figure 3.7

Chapter 3

Experimental design

The methodology used to build and exploit the Faster RCNN will be detailed in this chapter. This methodology follows as close as possible the methodology proposed in [24]. Finally, the results obtained with this protocol will be discussed in the Results chapter. The different technical aspects and the intermediate steps required to obtain the final results will be detailed.

3.1 Methodology

The goal of the methodology is to observe the detection performance between two Faster RCNNs with different depths. The methodology proposes to improve a shallower Faster RCNN with the integration of prior information. This information contains the past detections when a MAV swarm flies multiple times over the same field portions.

First, two Faster RCNN with different depths backbones (FCN5 and FCN9) are trained. The training is carried from scratch on simulated field portion images. Then, the two Faster RCNNs are evaluated on the evaluation dataset. A first comparison is made to see the classification performance by the network depth. In other words, if the Faster RCNN with an FCN5 backbone could be improved compared to the Faster RCNN with an FCN9 backbone.

If there is a performance gap between the networks, the prior-information (probability map) are integrated. The past detections computed by a Faster RCNN with a FCN5 backbone are used to build the probability map. Then, a Faster RCNN with a FCN5 backbone is trained with the training dataset. Finally, each image in the dataset is coupled with its probability map.
This training is done with different occurrence rates of the probability map in the images. The goal is to simulate the case when there are no past detections for a field portion.

The two Faster RCNNs are evaluated on the evaluation dataset. This dataset represents 10 multiple passages of a MAV over the same field portions (detailed in subsection 4.5.4). For each passage, the probability map of a specific field

portion is computed with the last detection made for this field portion. Thus, this strategy simulates successive passages of swarm MAV over the same field portion.

A separate evaluation is carried out for each Faster RCNN with a FCN5 backbone. Indeed, this Faster RCNN is trained with different probability map appearance rates. The goal is to compare the detection performance of this network by the map appearance rate.

Finally, the evaluation results are compared to see if the prior-information improve the detection performance of the Faster RCNN with a FCN5 backbone compared to the Faster RCNN with a FCN9 backbone.

In short, the methodology can be resumed in these steps:

1. Construction of two Faster RCNN: respectively with a FCN5 and FCN9 backbone.
2. Training of the two Faster RCNN on a simple evaluation dataset.
3. Comparison of the results to see if the network with a FCN5 backbone can be improved.
4. Generation of the probability map dataset based on the detection performance with a shallower Faster RCNN (FCN5 backbone).
5. Training of the Faster RCNN with a FCN5 backbone on the field images coupled to their probability map.
6. Evaluation of the both Faster RCNN across 10 successive passages over the same field portions by the map appearance rate.
7. The evaluation results obtained by a Faster RCNN with a FCN5 backbone are compared to those obtained with a FCN9 backbone.

3.2 Constraints and limits

The realization of this work was carried out by respecting as well as possible the methodology of the article. However, the totality of the parameters of the article could not be gathered within the range of this work.

3.2.1 Images and dataset

The original protocol uses a dataset composed of 500 images for the training step and 400 for the test step (the details of these datasets will be detailed in a dedicated section). The authors of the article [24] have kindly shared a part of this dataset to realize this work.

The simulator used to produce the original dataset has been largely modified since. It is therefore impossible to reproduce the original complete dataset. The simulator and the methodology used to obtain the data will be presented in the Image section. However, the dataset used in this work to reproduce the protocol corresponds to the original evaluation dataset in [24].

3.3 Technical setup

This section details the technical and software specifications used to perform this work.

3.3.1 Hardware

The hardware used in this work is designed to support fast calculations on images (multi-dimensional arrays). The Graphical Processing Unit (GPU) performs all the array calculations instead of the Central Processing Unit (CPU). The CUDA (Compute Unified Device Architecture) technology developed by NVIDIA is used to transfer all the array calculations to the GPU instead of the CPU to obtain faster computations.

The training of the different networks is completed with an NVIDIA RTX3090 graphic card. This graphic card has 10496 CUDA cores and a memory of 24GB [29]. Also, the RTX graphic card family has hardware optimizations for Tensor calculations. A Tensor is a mathematical generalization of the vectors used to represent the images and their annotations. The presence of 328 Tensor cores for the RTX3090 offers fast training cycles.

3.3.2 Software

This work has been developed with the Python3 language [43] and the Python package Pytorch [31]. The various networks were trained and manipulated with the version *1.0.8* of Pytorch as well as the version *11.1* of CUDA with the operating system Ubuntu *20.04LTS*.

3.4 Faster RCNN architectures

This section details the Faster RCNN used to perform the sugar beets and weeds detection. A Faster RCNN is divided into two parts: the convolutional neural networks to produced features maps (backbone) and the head. Two different Faster RCNN with a different backbone are built:

- A FCN5 backbone with 5 convolutional layers
- A FCN9 backbone with 9 convolutional layers.

The architectures of the backbones are presented in the Figure 2.1. The head of the Faster RCNN contains the Region Proposal Network (RPN) to detect the region of interests (ROIs) on the feature maps. Finaly, classification and regression layers predict classified bounding boxes.

3.4.1 Region Proposal Network

The RPN takes the feature maps sent by the backbone to classify and regress high-interest areas. First, the RPN predicts a series of region proposals based on the feature maps sent by the backbone. The size of the anchors generated by the RPN at each position in the feature map are 32x32, 64x64, 128x128, 256x256. This size range covers multiple parts of the feature map. The size (32x32) is defined to highlight the smallest elements. Also, the size of 256x256 covers the entire feature map and captures the more significant features.

The aspect ratio of the anchors is defined with three possible ratios 0.5, 1, 2. In short, the size of the anchors varies from 16x16 for the smallest to 512x512 for the largest. An anchor is positive if its overlap with a ground-truth object bounding box is greater than 0.7 and is negative if it is less than 0.3. First, the anchors are classified using a convolution layer to determine if the anchor frames an object or not. Then, the coordinates of the anchors are refined. A convolution layer will regress the 4 coordinates of the anchor to frame the object as precisely as possible.

3.5 Image

3.5.1 Origin

The images are simulated with the Unity 3D game engine. A sugar beet field has been simulated in the simulator and weeds were added. The images used in this work represent a portion of the simulated field with a picture taken at 3m altitudes.

Using a 3D simulator to simulate a field is very useful. Many parameters can be easily modified and control like the luminosity, altitude, plant species, soil types... Also, the Unity 3D game engine produces an automatic segmentation of the simulated scene. Segmentation is the highlighting of different elements in an image. In other words, the segmentation draws bounding boxes around the weeds and the sugar beets directly. For a simulated field portion (Figure 3.3), the annotations are automatically generated. Also, the segmentation generates a segmentation mask (Figure 3.5). A segmentation mask is a matrix where the value of each pixel is equal to the class belonging to the pixel. In Figure 3.5, the weeds pixels have a red value, the sugar beet pixels have a green value and the other have a black value (not assigned to a class).

The simulated images try to be as realistic as possible. However, acquiring images from real fields is a very long and costly process. Cover a field with drones is costly. Also, an automatic segmentation is not possible and the images have to be segmented manually. Variability is added to the simulation to minimize the reality gap with the simulator (random lights changes).

3.5.2 Plant constructions

A simulated sugar beet (Figure 3.1) is made of several leaves distributed around an axis located in the center of the plant. This axis has been slightly modified

and is divided into two levels. Each level represents a level of maturity of the plant. The leaves are distributed in a homogeneous way on each level. A disturbance is also applied to the position of the leaves.
The plants have a unique appearance while forming a coherent set in the field. Each image can have up to 80 sugar beets and up to 20 weeds (Figure 3.2). Each one represents a specific part of the simulated field. The different plants are placed on two possible soil textures. These textures are obtained with a Perlin Noise. This technique produces reproducible random and coherent textures with natural relief in an image.

Figure 3.1: Simulated sugar beet

Figure 3.2: Simulated weed

3.5.3 Structure

The goal is to simulate the images shooting by different MAVs. Also, MAVs fly at different times of the day and from different points of view. Each portion of the field is duplicated by undergoing changes and perturbations.

Variations in intensity and luminosity position are applied on the original portion. These changes modify the effects of shadows on the different plants. In Figure 3.4 these changes have been applied on an original portion (Figure 3.3) to produced the modified portion (Figure 3.4). In the original dataset, each field portion was duplicated 9 times. At each duplication, a different perturbation is applied to the original portion. These perturbations are organized in blocks. Each block has a modified copy of the original field portion. Also, each copy has a distinct modification. Thus, with this strategy, each block contains images with unique perturbations.

The Figure 3.3 illustrates the configuration of a crop segment. The plants are aligned in rows and weeds are visible between these rows like a real field. However, weeds are also placed in the crop rows to simulate a more realistic weed pattern. In a natural field, weeds can be placed everywhere. Multiple portions of the simulated fields are extracted to form the dataset.

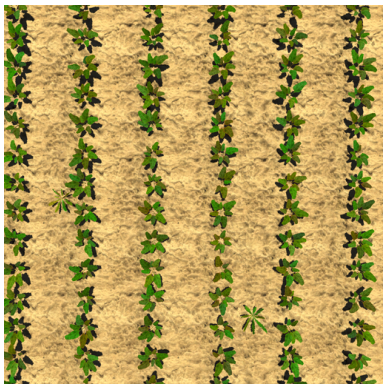

Figure 3.3: Portion of field A

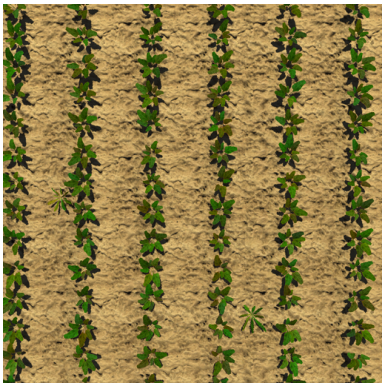

Figure 3.4: Modified portion of field A

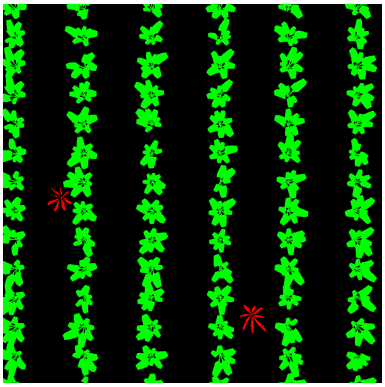

Figure 3.5: Ground truth segmentation for A

3.6 Dataset

The dataset contains all the images of sugar beet crops and weeds used to train the Faster RCNN to obtain efficient and precise predictions.

3.6.1 Composition

The dataset is formed with three elements: images, annotations and probability maps. Regarding the images, there is a total of 400 images. Each image represents a simulated field portion. Each image (PNG format) is in color (Red, Green, Black format) with a width of 512pixels and a height of 512pixels.
Each image has an annotation file. The annotation corresponds to the features information (weed and sugar beets) segmented by the simulator and is called the ground truth information. The annotations are stored in the form of a text file. A feature of the image is a set of crucial information and contains the class of the feature (weed or sugar beet) and the coordinates of the segmented bounding box in the format (x1,y1,x2,y2).

In a second step, the probability map is computed for each images. The map uses the predictions of the Faster RCNN with a 5 layers backbone. A new channel is added to the original RGB image to encode the probability map. This augmented image with 4 channels is a Numpy array stored with .NPY file extension. The Numpy Python library is handy to handle matrix operations.With Numpy, a probability map can be quickly added or removed in the last channel of an image according to a specific feeding rate of the probability map into the Faster RCNN.

3.6.2 Training set

The training set is composed of the images and their annotations. In some cases, the corresponding probability map is added to the images. The size of the training set is fixed at 330 images.

3.6.3 Validation set

The validation set contains all the images used to evaluate the performance of the Faster RCNN. Its size is fixed at 70 images.

These 70 images are distinct from the training set. They are new unseen images for the network to carry a proper evaluation of the performance. As explained previously, the validation set contains 7 unique plant patterns and all their variations in brightness. The validation set contains 70 images and is divided into 10 blocks of 7 images each.

3.6.4 Validation Strategy

The validation strategy defines the methodology to follow to carry an evaluation linked to the objectives of the detections. In this work, the detection task is to detect weeds and sugar beets.

Naive strategy

A basic and common evaluation strategy has been applied for a first time. This naive evaluation strategy consists of feeding the Faster RCNN with the whole evaluation set. Then, the Faster RCNN performance is observed.

This strategy took 50 random images from the dataset. Then, the evaluation is carried with a random shuffle of these images. This method is not robust. The images in the dataset are too similar. The different perturbations applied on each source field portion are not strong enough to bypass the training and validation data redundancy. In practice, the model validated images too close to the images used for training. This results in overestimated results.

In practice, the Faster RCNN optimizes all its weights to fit the training images during the training phase. The evaluation images are too similar to the training images and this similarity is not suitable to evaluate the generalization of the Faster RCNN detections appropriately.

In addition, the evaluation set structure is not compatible with the detection task. The evaluation set is completed with random images. This randomness of the data does not consider the multiples passages of a MAV over the same zone.

Current strategy

The current evaluation strategy avoids naive strategy issues. More images are added to the evaluation dataset and its structure changes. The evaluation dataset is divided into 10 blocks of 7 images each. The first block contains 7 unique images of a specific portion of a field. The other blocks (from blocks 2 to 10) contain these 7 images but with illumination changes. In short, each image in the original blocks has 9 different modified copies distributed on each block. This distribution along the blocks simulates the multiple passages of a MAV over the same zone. The illumination changes simulate the daylight variations and the forecast changes across time. An example of this distribution is shown in the Figure below.

- *light_changes*: is the illumination change function. This function takes a field portion image and returns this image with illumination changes applied to it (light source, light intensity...).
- *unique*: asserts that each transformation of an original field portion to a modified copy is unique. There are no two similar copies of the same field portion.
- n: the evaluation block number $2 <= n <= 7$.

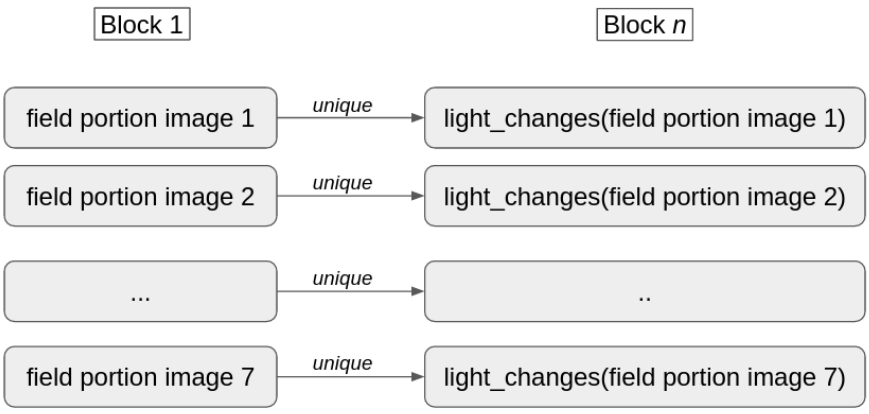

Figure 3.6: Structure of the evaluation blocks

To obtain a robust validation 7 source portions of the field were chosen. These portions are complex because they include all possible cases:

- Large overlap between sugar beets and weeds.
- High concentration of weeds in a small area.

- Weeds outside the crop lines.

The 7 portions of the field and their associated illumination variations(the 10 blocks) form the validation set. This method validates the detection task and validates the model on unseen images.

3.6.5 Size Ratio between Training and Validation set size

In [24], the evaluation dataset (400images) represents 44% of the total image volume (900).
The size of the dataset available for this work is only 400 images. It is not possible to respect the same proportion. Moreover, the number of weeds per image is low. Therefore, it is crucial to maximizing the number of training images to classify weeds efficiently. Indeed, an unbalanced class ratio in the dataset can lead to some issues and evaluation performance bias.

The training dataset should have 224 images to respect the same proportions. This number is too low and could compromise the classification efficiency. The training dataset will have insufficient numbers of images to train the Faster RCNN correctly. The size of the evaluation dataset is set to 70 images to maximize the weed detection performance.

3.7 Training

The training step is the minimization of the loss function. A training is carried over iterations or epochs with the images of the training dataset. As explained in the previous section, the training step was conducted with two different frameworks. The first framework used is the TorchVision library and the second is the maskrcnn-benchmark by Facebook. Therefore, the details provided in this section are independent of the framework used to perform the training. Only the metrics used to compute the length of the training step depend on the framework (epoch for TorchVision and iteration for maskrcnn-benchmark).

3.7.1 Preprocessing operations

Data preprocessing techniques have been applied to the training data set. These techniques prepare the raw loaded images to be fed in the Faster RCNN. An image is not fed to the network directly. The pixels of the raw image generally are integers between 0 and 255.

To increase the training efficiency, the input pixel values are normalized to minimize the value range. Normalize is transformed an image represented with a $[0, 255]$ range to a $[-1, 1]$ range without losing information. The goal is to obtain a pixel value mean = 0 and standard deviation = 1. First, the mean and the standard deviation of the pixel values for each channel in the images are computed. Then, the following computation is applied on each pixel P belonging to the channel i with the $mean_i$ as the mean of all the pixel value of the channel i and std_i as the standard deviation of all the pixel value of the

channel i :

$$P_i = \frac{(P_i - mean_i)}{std_i}$$

.

3.7.2 Data augmentation

Data augmentation is a set of modifications applied to the images. These modifications can be used to increase the size of the dataset. The modified images are added to the dataset. This method is handfull when the training images are scarce. Also, data augmentation can be used to introduce perturbations and noises in the images to prevent an overfit issue.

Image perturbations have been applied to the training image. The images are flipped with a random angle rotation. This transformation breaks the vertical aspect of the crop lines in the images. It also helps to simulate a more realistic image taken from a MAV. Indeed, a MAV can be misaligned to the crop lines and output a picture with a diagonal aspect of the crop lines.

3.7.3 Dataloader

A data loader is used to avoid loading the entire dataset in memory. A dataloader loads in memory only the necessary data necessary for the current iteration in the training phase. In concrete terms, without a dataloader, 370 images must be loaded in memory during all the training time. With a dataloader, there are only the number of necessary images (equal to the batch size) in memory to perform the iteration. For example, if the batch size is set to 4, the dataloader loads only the 4 required images.

In the context of this work, loading the entire dataset does not affect the performance. However, a dataloader has been implemented to have a scalable training. A scalable training will be helpful to enhance this work mainly if bigger datasets are used.

3.7.4 Training parameters

Concerning the Torchvision library, the training is performed on 10000 epochs. For the maskrcnn-benchmark framework, the training is performed on 50000 iterations. However, some parameters are common to both approaches:

- Learning rate: 0.01.
- Weight decay: 0.0001.
- Batch size: 4 images.

.

Due to some hardware computational constraints, some trainings have been done with smaller batch sizes. The training parameters are linked to the batch size when the training is done in the maskrcnn-benchmark framework. To keep these parameters consistent, they are divided by 2 if the batch size is divided by 2.
These parameters follow those defined in [24] to train the Faster RCNN in the same conditions.

3.7.5 Training strategies

Two types of trainings are carried out in this work. The first training type is set to train Faster RCNN on RGB images. Then, a second training is set on 4 channels images (with prior information).

1. FCN5 and FCN9 training on 3 channels images (RGB).
2. FCN5 training with 4 channels images (RGB + probability map).

Training on 4 channels images

Unlike the first type of training, a channel has been added to the training images. This channel corresponds to a probability map representing the weed detections of the previous passes of a MAV over this area. The details of the probability map will be detailed in the next section.

To improve the performance of a Faster RCNN with a five convolutional layers backbone, the previous detections made by this network will be added to the three existing channels of the input image. However, the map will not be added every time during training. Therefore, the network has to predict accurate detection even when there is no available prior information. To balance the weight of this new channel in the detections, three Faster RCNNs with a 5 layers backbone are trained with a varying probability map appearance ratio. The probability map appearance ratios during training are respectively: 25%, 50% and 75%. These three ratios are introduced to evaluate the performance changes of the Faster RCNNs when the prior information are present. The performance of the networks trained with these ratios will be presented in the result section.

Detections

The detections made by a well-trained Faster RCNN are illustrated in Figure 3.7. The red bounding boxes are linked to detected weeds and the green boxes to sugar beets.

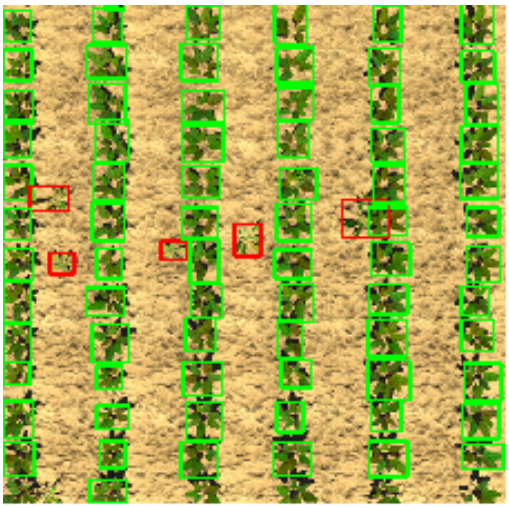

Figure 3.7: Faster RCNN detections on an field portion

3.8 Evaluation

The evaluation strategy objectively measures the detection performances of a network on new images. This evaluation is based on the analysis of 3 score metrics: precision, recall and F1 score.

The evaluation part is carries on a distinct dataset and the images are not shared with the training dataset. The evaluation dataset is composed of 70 images in total and includes 7 portions of unique fields.

The evaluation protocol is different depending on the evaluation goal. The first goal is to evaluate the detection performance of two Faster RCNN only RGB images. These two Faster RCNN have a different backbone depth, respectively a 5 layers and 9 layers backbone. For the first goal, the score metrics are computed based on an evaluation on the whole evaluation dataset. At this point, there is no need to handle the successive passages of MAVs over the same zone.

3.8.1 Evaluation on 4 channels Faster RCNN

A second goal is to evaluate the performance of the detections of Faster RCNN with a FCN5 backbone when prior information are added to the input image.

Indeed, this evaluation protocol must represent successive passages of MAVs over the same area. These successive passages taking place at different moments in time. The images of the evaluation dataset received random variations of luminosity. The objective is to simulate the changes in the environment at different moments in a day. As a reminder, the evaluation dataset is made of different 10 blocks of images. Each block contains the same plant pattern. The images on each block were modified with random illumination and brightness changes.

The evaluation protocol evaluates the detection performance of a Faster RCNN with a 5 layer backbone during 10 successive passes of MAVs over the same zone. Each block will represent a single MAV passage and is evaluated independently with three score metrics: precision, recall and F1 score. The evaluation scores obtained for each image inside the block are averaged:

$$Precision_{block} = \frac{\sum_i Precision(img_i)}{7}$$

$$Recall_{block} = \frac{\sum_i Recall(img_i)}{7}$$

$$F1block = \frac{\sum_i F1(img_i)}{7}$$

The detections made for the evaluation in a block are stored and used to compute the probability map to perform the images evaluation of the next block.

The evaluation protocol is illustrated by the figures below. Figure 3.8 illustrates the case when a MAV flies over a field portion for the first time. First, the MAV takes a picture from the field portion. No prior-information is available for the first passage, and the fourth channel of the image is left empty. The embedded Faster RCNN (FCN5 backbone) detects the weeds and the sugar beets on the portion. Then, the probability map of the weeds is computed. Figure 3.9 presents the case when a MAV of the swarm covers a zone for the second time. The MAV takes a picture from the field portion and sends this picture plus the probability map computed on the first previous passage to the embedded Faster RCNN. The sugars beets and weeds are detected and a new probability map is computed.

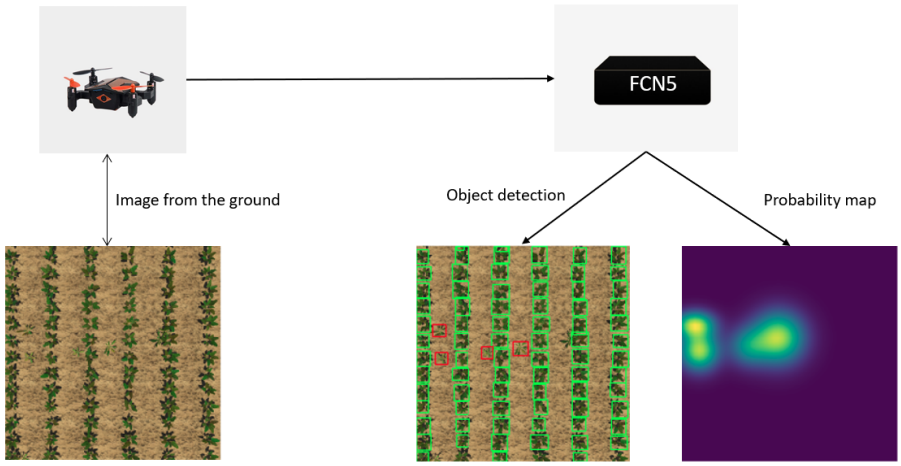

Figure 3.8: First passage of a MAV over a field portion

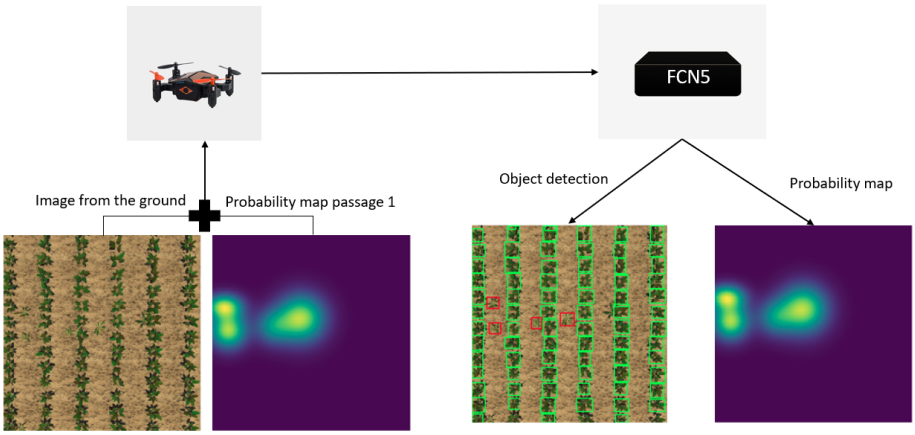

Figure 3.9: Second successive passages of MAV over a field portion

Note: the probability map of the first passage and the second passages are the same on the figures. The impacts of the probability map on the detections will be discussed in the Result chapter.

Classification score variations across blocks

The different brightness and illumination changes made in an image create variations in the classification performance of the network (Figure 3.10).

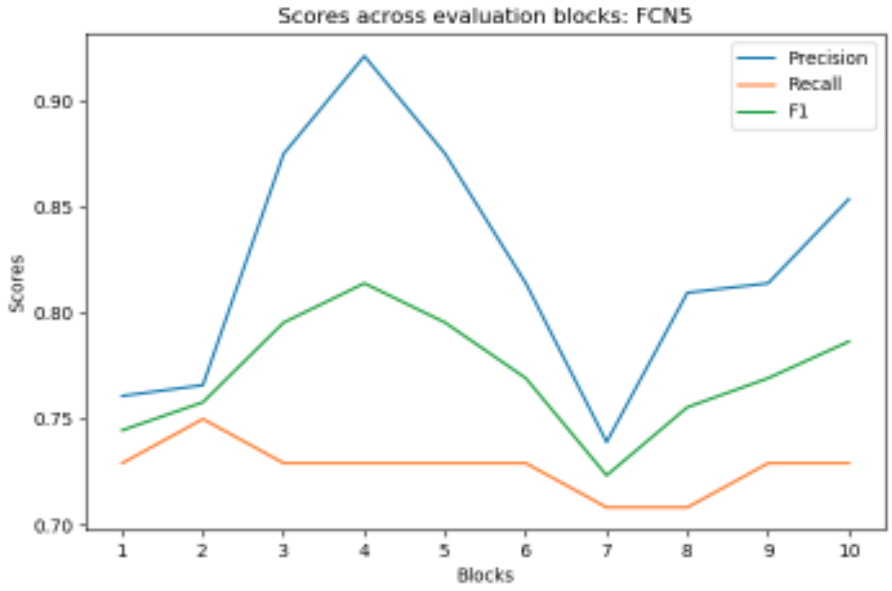

Figure 3.10: Evaluation scores across the different evaluation blocks of a Faster-RCNN with a FCN5 backbone

The network detection scores are strongly influenced by the block used for the evaluation. This could be explained by the low number of images present in

each block. In addition, the order of the blocks influences the evaluation results over 10 runs and introduces a bias in the evaluation strategy.

To bypass this bias, the order of the blocks in the evaluation sequence will be divided into sequences. A sequence corresponds to the evaluation of 10 blocks by the MAV swarm. For each sequence, the apparition order of the blocks is swapped to form a unique sequence. There are thus a total of 100 sequences corresponding to all the permutations of the appearance orders of the 10 blocks. In other words, a sequence defines an order of passage over the field portion.

To properly measures the scores over the block sequences, the scores are presented by passages. The score obtained by each passage is averaged across the sequences. The score for each metric is computed as followed:

$$Precision_i = \frac{\sum_{z=1}^{100} Precision(Sequence_z[i])}{100}$$

$$Recall_i = \frac{\sum_{z=1}^{100} Recall(Sequence_z[i])}{100}$$

$$F1_i = \frac{\sum_{z=1}^{100} F1(Sequence_z[i])}{100}$$

where :

- i: a specific MAV passage ($1 <= i <= 7$).
- $Precision_i$: the precision score for the i-th passage.
- $Precision_z[i]$: the precision score for the i-th block in the z-th sequence.
- $Recall_i$: the recall score for the i-th passage.
- $Recall_z[i]$: the recall score for the i-th block in the z-th sequence.
- $F1_i$: the F1 score for the i-th passage.
- $F1_z[i]$: the F1 score for the i-th block in the z-th sequence.

For example, the second block score of each sequence is averaged to obtain the evaluation performance of the second passage.

3.9 Probability map

The probability map of an image (representing a portion of the field) stores the prior-information gathered with multiple previous MAVs passages over the same zone. This probability map is a matrix with the same height and width as the original image. Each pixel value in the map is equal to the probability of being an element of the Weed class. The probability map development is detailed in the State of the art chapter in a dedicated section.

This probability map adds an *history* of the previous detections operated by the members of the Swarm. In [24], this probability is coupled to the original image of the field. The article also demonstrates an increasing detection performance when a shallower Faster RCNN (FCN5 backbone) uses a probability map. The results obtained by the network when the probability maps are used will be detailed in the Result section.

The probability map developed in this work is based on the same definition and methodology as exposed in the original article. These maps are the results of the detections made by a shallow Faster RCNN (FCN5 backbone) trained on 50000 iterations with RGB images. The probability map integrates all the detections belonging to the Weed class. Also, the detections are not filtered based on the confidence scores. If the confidence score is low, the detections could contain many misclassified elements. This absence of filtering is helpful to obtain a more realistic probability map with errors in it. Another Faster RCNN with a FCN5 backbone is trained with these unfiltered probability maps to deal with its errors. An example of a probability map is shown in Figure 2.7.

3.9.1 Limitations

The probability map in their original forms is computed only with the results of the current detection. This computation method does not take into account the past probability maps already computed for the same area. This method contains a disadvantage. The probability computation process is too sensitive to unanticipated wrong predictions of the network. If the network produces a lot of false positive weed detections for an undetermined reason, the probability map computed on these detections will contain all the false information. Therefore, the past expected correct detections cannot be used to counterbalance these wrong detections.

3.9.2 Probability map improvements

This work proposes an approach to circumvent the sensitivity problem when the previously computed maps are erased. This approach proposes to make a simple average between the map used to generate the prediction and the new computed map. The averaged probability map is defined in the following formula.

$$Pmap^{*}_{p+1} = \frac{Pmap_{p+1} + Pmap_{p}}{2}$$

Where:

- $Pmap_{p}$ is the probability map computed for the p-th passage of a MAV over the same zone.
- $Pmap_{p+1}$ is the probability map computed for the $p+1$-th passage of a MAV over the same zone.
- $Pmap^{*}_{p+1}$ is the new balanced probability map. The Faster RCNN will use this map to make detections for the next passage ($p+2$-th passage) of a MAV over the same zone.

This process aims to consider the previous map by giving equal weights between old and new detection. Thus, if the network predicts false results very different from the past detections, the old correct detections would still be kept in the new map. Also, the new erroneous detections will have less weight.

Under two assumptions:

- The previous detection matches the expected network performance.
- The wrong current detection is a rare and uncommon event.

This method would have the following advantages:

- Erroneous detections would have less weight and less influence on the future prediction of the Faster RCNN.
- Similar detections would have the same weight and so the same impact.
- Correct detections would be kept in the probability map and can continue to impact the future predictions of the Faster RCNN positively.
- The value of the correct detections would increase with each passage and the value of false detections would decrease.

The Faster RCNN detection performance obtained with the probability maps computed on both techniques will be discussed in the Result section.

Chapter 4

Results

In this chapter, the performances of two Faster RCNN according to their number of convolution layers (FCN5 and FCN9 backbone) will be compared. The comparison of their performances will first be developed on simple RGB images. Then, in a second step, their performance will be detailed on images integrating prior-information under the form of feature maps.

4.1 Network performances on RGB images

4.1.1 Score comparison with a weak evaluation strategy

A first training of 5000 epochs was conducted with the TorchVision libraries and evaluated with a weak evaluation protocol (70 random images). The results are compared between two Faster RCNN, one with a 5 convolution layers backbone (FCN5) and the other with 9 convolution layers (FCN9). The different scores were set according to the object's class to improve the readability. Also, the network performances on a specific class could be compared. In the following tabular, the class containing the sugar beets is called Crop and the class containing the weeds is called Weed.

In Tabular 4.1, the Faster RCNN with both backbones presents almost equivalent performances for the two classes. The score related to the Weed class is slightly lower than that of the Crop class due to two factors:

- Number of sugar beets > number of weeds in images. There is thus an unequal distribution of the two classes. The network has more training elements to classify sugar beets than Weeds.
- line crops in the images respect a well-defined geometrical pattern: plants are arranged in continuous lines. In comparison, weeds do not have this geometrical pattern. A weed can be present everywhere, between the lines or next to a plant. The geometrical pattern is generally a geometrical structure that helps the convolutional network to classify the elements better.

	FCN5		FCN9	
	Crop	Weed	Crop	Weed
Precision	0.91	0.96	0.91	0.88
Recall	0.91	0.87	0.92	0.89
F1	0.91	0.87	0.92	0.89

Table 4.1: Faster RCNN evaluation scores by class and by backbone

4.1.2 Score comparison with a strong evaluation strategy and more training iterations

The Faster RCNN with a 5 convolution layers backbone has been re-trained on a new improved training and evaluation dataset. The new evaluation dataset contains 7 unique plant patterns and their variations for a total of 70 images. The new training dataset contains the 330 remaining images. The number of epochs has also been increased. The goal is to see if the performances can be improved by getting closer to the number of iterations defined in the article to reproduce the protocol.

Two Faster RCNNs with a FCN5 backbone have been trained with this new training dataset. The first one was trained on 5000 epochs and the second one on 10000 epochs (both with a FCN5 backbone).

	5000 epochs		10 000 epochs	
	Crop	Weed	Crop	Weed
Precision	0.92	0.86	0.96	0.95
Recall	0.96	0.58	0.91	0.60
F1	0.93	0.69	0.93	0.73

Table 4.2: Faster RCNN with FCN5 backbone strong evaluation scores by class and by epoch number

In the Tabular 4.2 the doubling of the number of iterations shows an improvement in the classification of Weeds while the classification of crops remains stable. This improvement in the detection of Weeds is supported by a strong increase in the precision of the network detection. There is also a very slight increase in the recall score. A higher number of training epochs can explain this increase. The Faster RCNN has more epochs to improve its weights and perform detection to detect weeds.
However, the recall scores tend to decrease with more epochs for the Crop class. The recall is already very high with a training on 5000 epochs. A training on 10000 epochs could lead to an overfit case for the Crop class. The overfit risk is very high especially when a nearly perfect precision score is achieved. This situation can lower the generalization of the network and increase the number of false-negative.

4.1.3 Limitations

The performance obtained by a Faster RCNN with a 5 convolution layers backbone, trained on 10000 epochs with TorchVision library and evaluated with a strong strategy shows higher results than the same Faster RCNN but trained on fewer epochs. However, the number of iterations announced in the paper [24] is 50000. Moreover, the doubling of the number of epochs is not efficient. As a result, the classification performance increased by only 0.05% for the Weed class and stabilized the performance for the Crop class.

4.1.4 Score comparison with 50000 training iterations

Based on these specifications, two Faster RCNNs have been trained over 50000 iterations with a batch size of 4 images with the help of the maskrcnn-benchmark framework. Each Faster RCNN has been trained respectively with a FCN5 and a FCN9 backbone. Moreover, these models have been evaluated with a strong evaluation strategy. Henceforth, the parameters between the Faster RCNN developed in this work and those developed in [24] differ only in the size of the training and evaluation dataset.

	FCN5		FCN9	
	Crop	Weed	Crop	Weed
Precision	0.99	0.82	0.99	0.90
Recall	0.97	0.73	0.97	0.78
F1	0.98	0.77	0.98	0.83

Table 4.3: Faster RCNN evaluation scores by class and by backbone with a strong evaluation strategy and trained on 50000 iterations

Regarding the results for the Weed class in Tabular 4.3, the F1 scores of the FCN9 backbone are higher than the FCN5 scores.
Faster RCNNs with both backbones achieve good detections with few false positive (high precision) but tend to detect a bit more false negative (good recall).

Regarding the performance on the Crop class in Tabular 4.3, the Faster RCNNs with both backbones classify more efficiently the elements of the Crop class than the elements of the Weed class. Furthermore, the precision and the recall are very high with nearly perfect scores. This performance can be explained with a reduced number of data and a high number of iteration. This context can lead to an over-learning of the elements of the Crop class.

4.1.5 Conclusions

Regarding the results, the Faster RCNNs developed in this work classify the Crop class efficiently independently from the depths.
However, the shallower Faster RCNN produces a less efficient classification for the elements of the weed class. This under-performance leaves space to improve the Faster RCNN with a FCN5 backbone to increase its classification scores for the weed class. Among these improvements, the introduction of prior-information from previous classifications made by MAVs for the same area.

4.2 Network performances on RGB image with Probability map

This section will present the results according to the feeding rate of the probability map in Faster RCNNs. In the first step, the probability maps are calculated only based on the current detections. The map does not take into account the previous computed probability map. In a second step, the probability maps will be computed with an averaging between the current probability map and the previous probability map used to produced the detections.

4.2.1 Scores comparison on original feature map calculation

Score metrics group the results, the precision, recall and F1 are used to measure the performance. The score of each Faster RCNN (FCN5 backbone) trained with a specific probability map feeding rate are compared against the score obtained by the Faster RCNNs trained without feature map (FCN5 and FCN9 backbones).

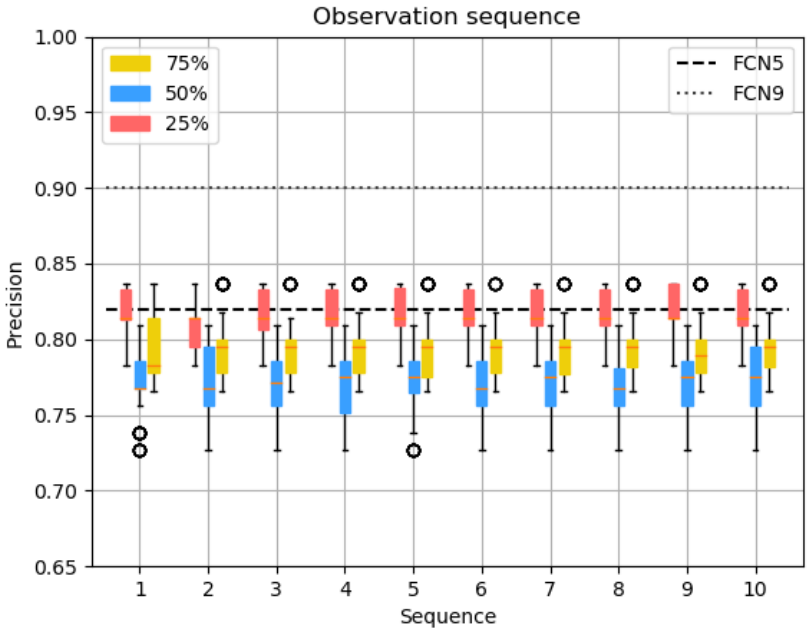

Figure 4.1: Precision scores for FCN5 and FCN9 backbones across passages and per probability map feeding rate.

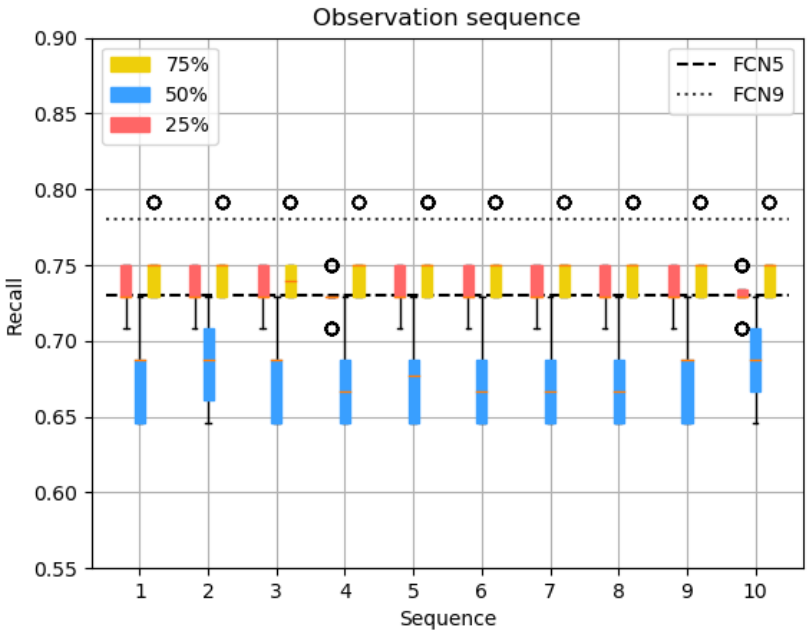

Figure 4.2: Recall scores for FCN5 and FCN9 backbones across passages and per probability map feeding rate.

It is clear that whatever the feeding rate, the addition of prior-information on the input images in the Faster RCNN (FCN5 backbone) does not improve either the precision (Figure 4.1 or the recall (Figure 4.2). Moreover, precision and recall are degraded when the network is trained with a probability map feeding rate of 50%. Moreover, the precision slightly decreases when the prior information are provided to the network with a rate of 75%. These trends impact the global performance of the network and impact the F1 score similarly.

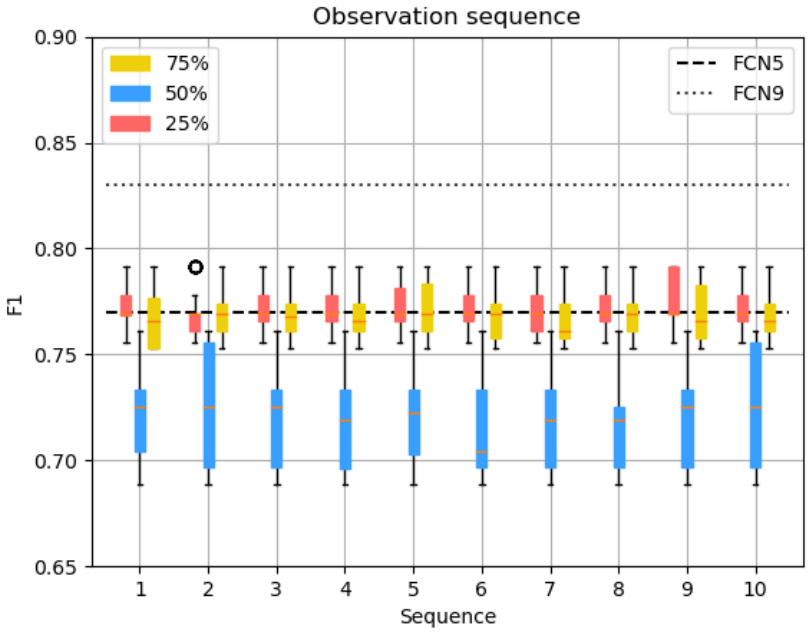

Figure 4.3: F1 scores for FCN5 and FCN9 backbones across passages and per probability map feeding rate.

Regarding the F1 score (Figure 4.3), the introduction of prior information in the network does not affect the F1 score and does not bring any improvement. Moreover, the score is getting worse when the feeding rate is 50%.

Prior information does not improve the overall performance of the Faster RCNN with a 5 convolutional layers backbone. Indeed, the scores do not increase across multiple passages. This stabilization of the performance shows that mixing past predictions with current predictions does not improve the network's ability to compute better detections. Instead, the Faster RCNN classifies equally well on each passage independently of the prior-information available.

4.2.2 Scores comparison on averaged feature maps

A better approach has been tried to improve the effects of the prior-information on the detection performance of a Faster RCNN (FCN5 backbone). This approach tries to limit the impact of erroneous information on the feature map to

improve future detections.

The results are presented in the graphs below and are structured similarly as the previous score graphs.

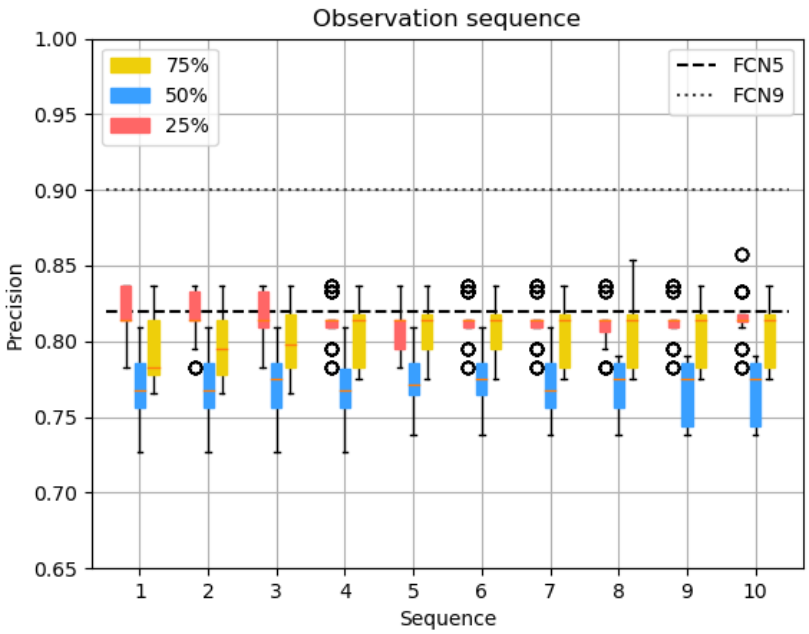

Figure 4.4: Precision scores for FCN5 and FCN9 backbones across passages and per probability map feeding rate.

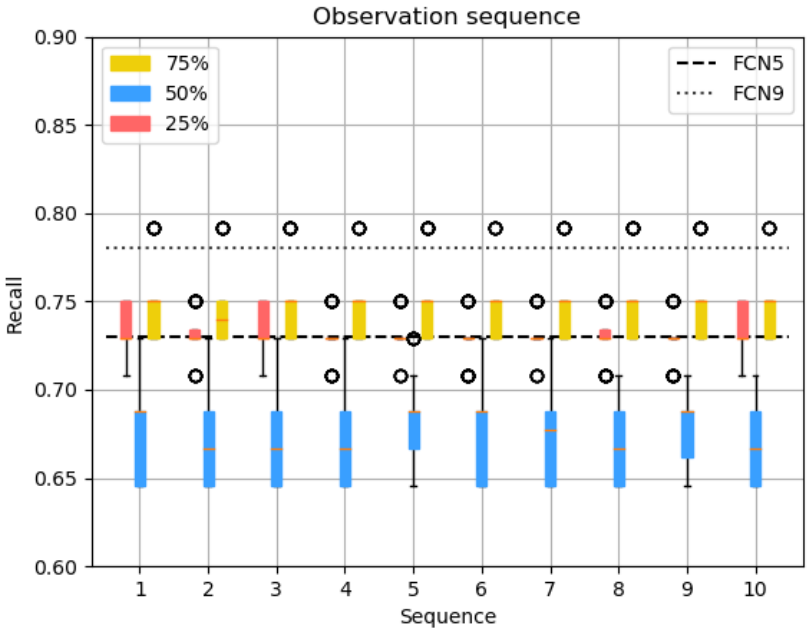

Figure 4.5: Recall scores for FCN5 and FCN9 backbones across passages and per probability map feeding rate.

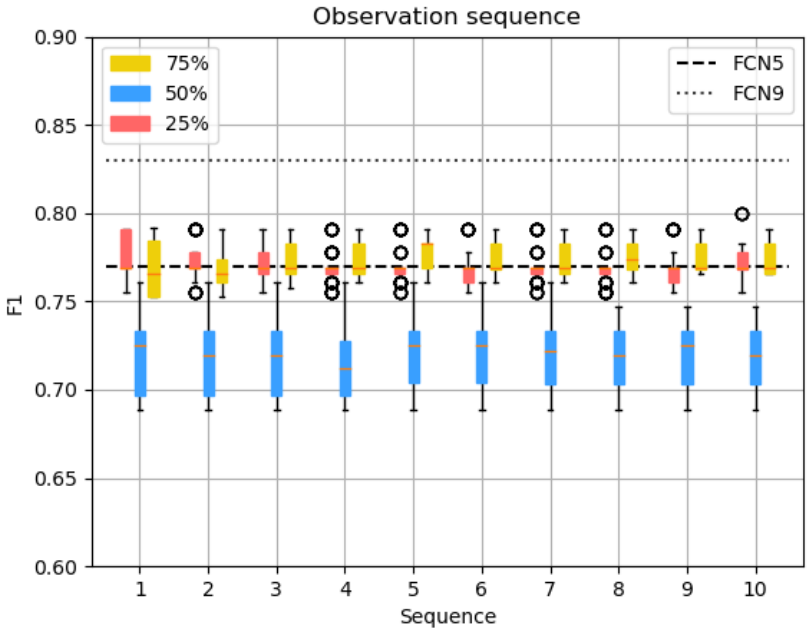

Figure 4.6: F1 scores for FCN5 and FCN9 backbones across passages and per probability map feeding rate.

As shown in Figure 4.6, the probability maps do not have a positive effect even if they are averaged. The F1 score of a Faster RCNN with a 5 layers backbone is not impacted. Regarding the precision (Figure 4.4), the performances are degraded when the feeding rate is equal to 50 or 75% compared to a Faster RCNN with a FCN5 backbone trained only on RGB images. For the recall (Figure 4.5), the performances are equivalent to the reference Faster RCNN (FCN5 backbone) if the feeding rate is not equal to 50%. When the prior-information are used one out of two times to train this network, the performances are decreased.

The feature map computation method does not affect the detection of a Faster RCNN with a FCN5 backbone. The averaged feature maps have the same performance results as the classic feature maps and no improvement is visible.

4.2.3 Conclusions

In conclusion, taking into account the prior-information related to the successive passages of a MAV over the same area does not improve the Faster RCNN performance under the conditions conducted in this work. In some cases, the prior-information even degrades the performance of the network.

The feature maps do not improve detections on a Faster RCNN with an FCN5 backbone. Under these conditions, using a shallower neural network with prior-information consideration does not match the performance of a deeper Faster

RCNN for detecting weeds. Therefore, using a Faster RCNN with an FCN9 as a backbone is more effective to detect weeds than a Faster RCNN with an FCN5 backbone to classify sugar beets and weeds by a MAV Swarm.

Chapter 5

Discussion

5.1 Discussion on the result gaps with the source article

This section details the differences in the results obtained in this work and in the article [24] by applying the same methodology.

5.1.1 Data amount

The dataset used in this work represents a fraction of the original dataset used in [24]. The dataset used in this work consists of 400 images in total and is divided as follows: 330 images for training and 70 images for validation. In [24], there are 900 images available and divided as follows: 500 images for training and 400 for validation.
Also, the article uses two different datasets to apply the methodology. The first dataset presented contains 900 field portion images (dataset A). The second dataset (dataset B) contains the same field portion but with camera position errors. The dataset B is not available and is not used in this work.
The results from the [24] presented in this section will be exclusively related to dataset A.

There is a heavy dataset size gap and this work tries to consider the decrease in the number of images. The risk of over-training the network is high. The network would learn by heart the training data and lose efficiency on new images (overfit case). However, the reduced size of the dataset is sufficient to apply the methodology. The impact of the lack of images will be considered in the development of this work.

5.1.2 Results on RGB images

There are differences between the results produced in this work with Faster RCNNs trained on 5000 epochs (Tabular 4.1) and those presented in [24] (Tabular 5.1).

The main divergence point between the two results are the evaluation strategy, the number of available images and the number of training epochs. Indeed, the dataset in [24] includes an evaluation dataset of 400 images. This dataset is divided into blocks of 10 images with different light variations.
The results of the first evaluation strategy used in this work are presented in Tabular 4.1. This strategy does not include the evaluation blocks system and has fewer images (only 70). Moreover, the Faster RCNN are trained with 5000 epochs (unlike 50000 iterations in [24]).

	FCN5		FCN9	
	Crop	Weed	Crop	Weed
Precision	0.96	0.98	0.96	0.98
Recall	0.87	0.65	0.91	0.73
F1	0.91	0.78	0.93	0.89

Table 5.1: Faster RCNN evaluation scores by class and by backbone in [24]

5.1.3 Strong evaluation strategy on RGB images

The performances between the Faster RCNN developed in this work and those developed in [24] show differences.

Performance on the Weed class

Regarding the performance on the Weed class in Tabular 4.3, the results differ depending on the type of backbone used.

Regarding the FCN5 backbone, the one developed in this work has a lower precision but a higher recall than in [24]. In other words, the model generates fewer false-negatives but more false-positives. This balance between false-negatives and false-positives leads to an overall weed classification performance equal to the Faster RCNN with a FCN5 backbone developed in [24] (Tabular 5.1).

Concerning FCN9 backbone, the mechanisms are similar. However, the false-negative and false-positive balance of the Faster RCNN with a FCN9 backbone is more unbalanced in this work and leads to lower overall performance (Tabular 4.3) of the Weeds classification than in [24] (Tabular 5.1).
This difference can be explained by the much lower number of elements of the Weed class used for training in comparison with [24].

Performance on the Crop class

The results for the Crop class obtained in this work are higher than those exposed in [24]. These results are valid for both Faster RCNNs (FCN5 and FCN9). The precision and the recall are nearly perfect (0.99 and 0.97) and could be explained by an overfit situation for the Crop class. More significantly, the recall has been increased in the Faster RCNN developed in this work. The Faster RCNN could have learned too much Crop class elements due to the smaller amount of data than in [24].

5.2 Faster RCNN frameworks and their impacts on the methodology

Firstly, the TorchVision library of the Pytorch project is used to implement the Faster RCNN. This choice was motivated by the fact that TorchVision isthe official library of the PyTorch project. Furthermore, these libraries are also well updated and well documented, making it easier to learn to use PyTorch and grasp the development of Faster RCNN.

5.2.1 Misunderstanding between epochs and iterations

The number of iterations defined in the article [24] is 50000 iterations. The framework not being specified, the first Faster RCNN were developed with the TorchVision library to reach 50000 iterations and obtain a similar environment to reproduce the protocol. However, confusion was made between iterations and epochs. Indeed, 50000 iterations with a batch size of 4 images are equivalent to 400 epochs. This number of epochs is much too low to train this type of network. Therefore, a first hypothesis was made by interpreting iteration as being equal to one epoch. The impacts of this hypothesis on the methodology are detailed in the subsection below.

5.2.2 Impact on the evaluation strategy

A first Faster RCNN was trained on 5000 epochs with the TorchVision library. The goal is to evaluate the Faster RCNNs with the 5 and 9 convolutional layers backbones. The results (Figure4.1 were encouraging but below the performances reached in [24] (Figure 5.1). A level of 10000 Epochs has been set to observe if the networks could obtain the desired performances. Also, it is interesting to see how the performance will be improved when doubling the epoch number.

With a training of 10000 epochs, the performance scores exceeded the scores obtained in [24]. Therefore, the evaluation protocol is questioned. The protocol does not take into account the previous passage of a MAV over an area. Also, the images used for the validation are too similar to the images used for the training.

A robust validation strategy has been developed to improve the evaluation strategy. The objective was to be sure that the validation dataset correctly measures the model's performance on unseen data. With this strong validation strategy, the performances (Tabular 4.2) were slightly below the paper ones (Tabular 5.1). With this strategy, the shallower Faster RCNN has performed better classification than the deeper Faster RCNN.

5.2.3 Reconsideration of the framework

One of the reasons that could explain this confusing result was the much lower number of epochs used for the training than the number of iterations defined in [24]. The Faster RCNN with a FCN5 as backbone could have converged faster due to its shallower depth.

The first solution to this problem is to increase the number of epochs again to get closer to the defined number of iterations. Increasing the epochs number could give more time for the Faster RCNN with a FCN9 backbone to converge and increase its performance. However, the training time becomes too high and unfeasible in practice. Training the networks with TorchVision libraries is very slow. More than 15h are required to complete a training with 10000 epochs. In the TorchVision library, an epoch during a training cycle is complete when the whole dataset has been given to the network. A discussion was opened with the authors of [24] to present them the problem. The authors used another framework, the maskrcnn-benchmark from Facebook Research `https://github.com/facebookresearch/maskrcnn-benchmark`.

5.2.4 Maskrcnn-benchmark

The maskrcnn-benchmark framework contains many optimizations to increase the training speed by decreasing the GPU memory needed to perform a training run. In this framework, an iteration is complete when the entire batch has been delivered to the network. The development of the Faster RCNN with TorchVision is stopped. Instead, the maskrcnn-benchmark is used to take advantage of its optimizations and minimize training time.

With this framework and a proper definition of an iteration, training with 50000 iterations could be realized in a reasonable time (+-7h) with batches of 4 images. The results obtained with this framework are more consistent with the methodology defined in [24]. These results will be discussed in a dedicated section.

5.3 Discussion on the impacts of an improved dataset

Two faster RCNN are trained and evaluated with a new dataset to observe a performance gap between the two networks. The goal is to circumvent the negative effects of the first dataset on the results and evaluate if the full methodology can be applied to this dataset.

The disadvantages and the weak aspects of the first dataset are identified and can be resumed in two points:

- Insufficient number of images.
- Unbalanced class element ratio: too few weeds in fields.

A new improved dataset has been generated to have a new training and evaluation dataset without these negative points. This new dataset has been kindly shared by Carlos Carbone (Sapienza Università di Roma).

Construction and origin

The simulator Unity 3D is used to simulate sugar beets fields. The simulation is based on images taken from real sugar beets crops to have more realistic images. A complete sugar beet field is simulated to take pictures of its different portions.

Unlike the first dataset used in this work, the weeds use in the new dataset are volunteer potatoes. Volunteer potatoes often appear in the case of field rotation. When potato tubers planted in the last season are left after the harvest. These potatoes can grow on the side of other plants and waste space and resources (water, soil nutrients...).

In the simulated field the sugar beets are placed in lines. Unlike the first dataset, there is only one stage to place the leaves and all the leaves of a sugar beet start at the same point at the root (Figure 5.2).
Regarding the potatoes (Figure 5.1), they are structured like small bushes of leaves. Potato stems are placed in the soil (with random noise). The leaves are added in two places along each stem also at the head of each stem.

Figure 5.1: Simulated volunteer potato

Figure 5.2: Simulated sugar beet

Compared to the first dataset, this dataset also has more sugar beet and weed elements per image. For example, in a field portion image taken at 20m altitude, there are around 150 sugar beets and 50 volunteer potatoes.

Structure

The dataset contains 1200 field portion images. In addition, some perturbations and noises are applied on 400 original field portions to create more images. These noises can be categorized into 3 levels and each level contains 400 field portions images. The levels are defined as follow:

1. Level 0: no perturbation. This level corresponds to the original field portion images(Figure 5.3).
2. Level 1: Random camera rotations (with an angle between 180 and -180°)are applied on each field portion to modify the point of view (Figure 5.4).
3. Level 2: Random camera rotations, illumination changes (illumination intensity and rotation) and different camera altitudes (3m, 10m, 20m and 40m from the ground) are applied on each field portion (Figure 5.5).

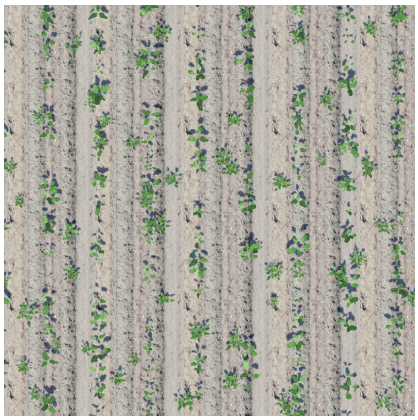

Figure 5.3: Field portion with noise level 0

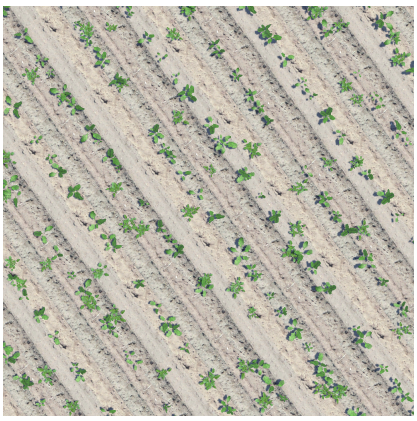

Figure 5.4: Field portion with noise level 1

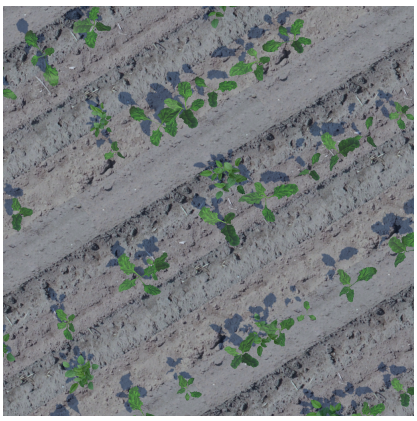

Figure 5.5: Field portion with noise level 2

The training dataset contains 260 images are randomly taken from each noise level for a total of 780 field portion images. Also, the evaluation dataset contains the 420 remaining images.

Results

Two Faster RCNNs with different backbones (FCN5 and FCN9) are trained on 50000 iterations with the maskrcnn-benchmark framework. The results of the evaluation (Figure 5.2) show no important performance difference between a Faster RCNN with a FCN5 or a FCN9 backbone trained on the new dataset.

The Faster RCNNs classify the sugar beets (Crop class) efficiently and there are few false-positive (precision of 0.93 and 0.92). The Faster RCNNs also well classify volunteer potatoes (Weed class) with a precision of 0.81 and 0.85. The precision score gap between the two classes could be explained with higher complexity of the potato shape (bush architecture) compared to the sugar beets (single root) and a higher number of sugar beets than the potatoes in the dataset.

However, the recall is very low for the Crop class (0.55) for both Faster RCNNs, especially in comparison with the Weed class (0.74). There are many false-negative detections, Faster RCNNs have difficulties detecting all the potatoes in a field portion. These results are intriguing because the recall for the Crop class is higher even if there are more sugar beets in images with lower shape complexity.

More generally, there is no performance gap between the backbones used in the Faster RCNN. The F1 scores are equal between FCN5 and FCN9 for each class.

	FCN5		FCN9	
	Crop	Weed	Crop	Weed
Precision	0.81	0.93	0.85	0.92
Recall	0.74	0.55	0.74	0.55
F1	0.77	0.69	0.79	0.69

Table 5.2: Faster RCNN evaluation scores by class and by backbone

5.3.1 Conclusions

Train two Faster RCNNs with different backbones on a new dataset with more images and more weeds shows no classification performance gap between the two networks. In these conditions, there is no space to bring improvements on the Faster RCNN with a shallower backbone (FCN5) with the prior-information taken from multiple passages. However, the performance of both Faster RCNNs are not perfect and can be generally improved.

Chapter 6

Future works

This chapter details the changes that could be implemented to improve this work.

6.1 Improved dataset

The data is a crucial point to improve the quality of the detections. A beginning of improvement has been introduced with a new improved dataset. Unfortunately, the results obtained with this new dataset are not sufficient to exploit the prior-information taken from multiple passages of a MAV over the same field portion. However, this work provides a solid basis to improve these results.

Until now, all the Faster RCNNs used in this work are trained with images generated from a 3D simulator. The second new dataset brings more realistic field images with perturbations to simulate a realistic image capture from a MAV over a field portion. However, they are always a reality gap between simulation and images taken in a natural field. It could be interesting to evaluate the Faster RCNN trained with simulated images on real images to measure this reality gap. One possible improvement is to train the Faster RCNNs with real images to see if the prior-information can be helpful to improve the detection inside a MAV swarm.

6.2 Mask RCNN

The Mask RCNN [15] is an extension of the Faster RCNN and adds the ability to classify each pixel in the image to create a segmentation mask. A segmentation mask is a raw image with the same dimension as the original image where each pixel color corresponds to a prediction of class membership. These masks could be used to train other types of state-of-the-art supervised learning CNNs such as U-Net [37].

Mask RCNN introduces a new branch parallel to the classification and regression branches. This new branch also takes an ROI and returns binary masks

from that ROI. A binary mask is a mask where the pixel color is a class membership. For example, all the red pixels are weed pixels. The unclassed pixels are black otherwise.

A binary mask produced by a Mask RCNN could be used to detect the sugar beets and weeds in the fields with improved precision. The disadvantage of the Faster RCNN is the precision of the bounding boxes. A bounding box frames at best an element. However, there are always useless pixels framed by the bounding box. Therefore, a pixel segmentation with a binary mask might provide a more precise segmentation of the plants in a field.

6.3 Framework

In this work, two frameworks were used: TorchVision library and the maskrcnn-benchmark. Facebook Research has developed a better version of the maskrcnn-benchmark: Detectron2 [46]. This framework proposes a faster training with a network feeding rate of 64 images/seconds (instead of 53 for maskrcnn-benchmark) and also adds more features like the support of rotated bounding boxes.

A classic bounding box frames the objects with horizontal and vertical parameters and can be represented in this way $[x, y, width, height]$ (where x and y are the lowest corner coordinates of the box). A rotated bounding box adds the parameter θ. This parameter represents the angle of the object with the vertical axis. In Figure 6.1, the rotated bounding box is more precise than the bounding box to frame an object placed in diagonal. Using a framework like the Detectron2 to implement the Faster RCNN could improve the overall training speed. Also, a more precise detection is possible when the plants have not a squared shape in a field.

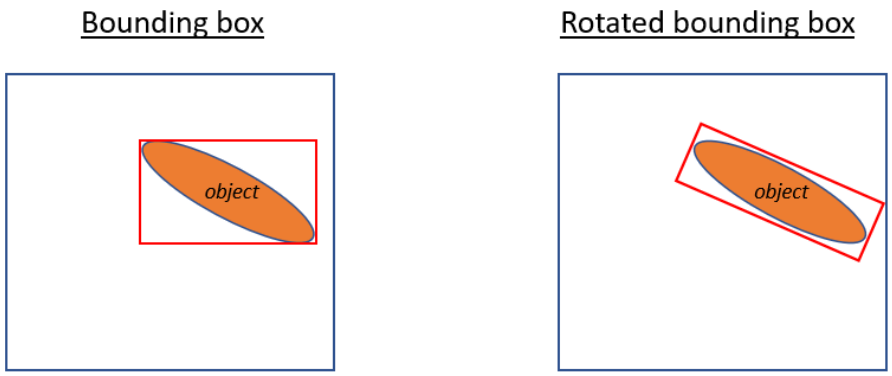

Figure 6.1: Comparison between bounding box and rotated bounding box

Chapter 7

Conclusions

In conclusion, the objective of this work was to introduce the reader to the Region based CNN throughout the problem of weed control in the context of precision agriculture. Efficient detection of weeds in the fields contributes to target the weeds in the fields better and reduces the volume of pesticides used. This work takes the methodology defined in [24] and reproduces it as faithfully as possible. The goal of the methodology is to observe if a shallower Faster RCNN can be improved compared to a deeper Faster RCNN when prior-information taked from multiples previous passage of a MAV swarm over the same zone are used.

In a first time, two Faster RCNNs with different depths were built. One Faster RCNN has a backbone of 5 convolution layers (FCN5) and the other has a backbone of 9 convolution layers (FCN9) to compute the feature maps. Then, the two Faster RCNNs are trained from scratch with a fraction of the dataset used in [24] to compare their classification performance gap. Multiple trainings are performed with different parameters and frameworks. The first training phases are performed with the TorchVision libraries with different epochs numbers (5000 and 10000). Finally, the maskrcnn-benchmark framework is used to train the Faster RCNNs on 50000 iterations. Performance differences have been highlighted between the two Faster RCNN.

In a second time, the prior-information resulting from the multiple MAV passages over the same zone were implemented. The objective is to observe the classification performance of the Faster RCNN with a FCN5 backbone when the prior-information are added. To realize this objective, the probability maps encoding the detections of the previous passage over the same zone are built. Then, a Faster RCNN with a FCN5 backbone is trained on the original images coupled with their respective features maps. Multiple trainings are performed with a different apparition rate of the probability map to see how the Faster RCNN will react with and without prior-information.
The results show no improvements in the classification results of the Faster RCNN when the prior-information are used (independently of the apparition rate).

After the methodology reproduction and the analysis of the results, this work developed a series of discussions. A first discussion based on the results obtained in this work and those obtained in [24] is proposed. This discussion detailed the differences between the two developments and proposes analyzing the reasons behind the results gap.
Then a second discussion is developed. This discussion explores the differences and the impacts on the methodology between the two frameworks used to build and train the Faster RCNN in this work.
A third discussion is proposed to observe the impacts on detection performance when the Faster RCNNs with different backbones are trained with a new improved dataset. This new dataset considers the defaults of the first dataset and contains more images with more realism. The evaluation results are the same for both Faster RCNNs but could be improved in further works.

This work provides a solid basis to develop and exploit the power of Faster RCNN in a concrete way. The Region Based CNN are complex and involve many abstract concepts. Through the reading of this work, the reader can have an overview of the architecture and the utility of this kind of CNN. The rigorous methodology developed along this work shows the Faster RCNN power and its weak points. This work also offers a scientific application of Faster RCNN on a concrete case with weed control management in precision agriculture.

Bibliography

[1] Livraisons par drone. https://www.dpd.fr/livraison-drone. Accessed: 2021-05-08.

[2] General Atomics MQ-9 Reaper. https://en.wikipedia.org/w/index.php?title=General_Atomics_MQ-9_Reaper&oldid=1016648992, April 2021. Accessed: 2021-04-14.

[3] Qassim A. Abdullah. Classification of the Unmanned Aerial Systems | GEOG 892: Unmanned Aerial Systems. https://www.e-education.psu.edu/geog892/node/5. Accessed: 2021-05-09.

[4] Dario Albani, Tiziano Manoni, Arikhan Arik, Daniele Nardi, and Vito Trianni. Field Coverage for Weed Mapping: Toward Experiments with a UAV Swarm. In Adriana Compagnoni, William Casey, Yang Cai, and Bud Mishra, editors, Bio-inspired Information and Communication Technologies, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 132–146, Cham, 2019. Springer International Publishing.

[5] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET), pages 1–6, Antalya, August 2017. IEEE.

[6] Isabel Cisternas, Ignacio Velásquez, Angélica Caro, and Alfonso Rodríguez. Systematic literature review of implementations of precision agriculture. Computers and Electronics in Agriculture, 176:105626, September 2020.

[7] Nan Cui. Applying Gradient Descent in Convolutional Neural Networks. Journal of Physics: Conference Series, 1004:012027, April 2018. Publisher: IOP Publishing.

[8] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255, June 2009. ISSN: 1063-6919.

[9] Yael Edan, Shufeng Han, and Naoshi Kondo. Automation in Agriculture. In Shimon Y. Nof, editor, Springer Handbook of Automation, Springer Handbooks, pages 1095–1128. Springer, Berlin, Heidelberg, 2009.

[10] Emilio Garcia-Fidalgo, Francisco Bonnin-Pascual, and Alberto Ortiz. A Control Architecture for a Micro Aerial Vehicle Intended for Vessel Visual Inspection. In Proceedings of III Jornadas de Computación Empotrada (JCE), pages 4–9. JCE, September 2012.

[11] Ross Girshick. Fast R-CNN. arXiv:1504.08083 [cs], September 2015. arXiv: 1504.08083.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv:1311.2524 [cs], October 2014. arXiv: 1311.2524.

[13] L.K. Hansen and P. Salamon. Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(10):993–1001, October 1990.

[14] S. Haug, A. Michaels, P. Biber, and J. Ostermann. Plant classification system for crop /weed discrimination without segmentation. In IEEE Winter Conference on Applications of Computer Vision, pages 1142–1149, March 2014. ISSN: 1550-5790.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. CoRR, abs/1703.06870, March 2017.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, Las Vegas, NV, USA, June 2016. IEEE.

[17] Jian Jin and Lie Tang. Corn plant sensing using real-time stereo vision. Journal of Field Robotics, 26(6-7):591–608, 2009. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.20293.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6):84–90, May 2017.

[19] Esther Levin, Naftali Tishby, and Sara Solla. A Statistical Approach to Learning and Generalization in Layered Neural Networks. Proceedings of the IEEE, 78:1568–1574, November 1990.

[20] Yang Liu and J. K. Aggarwal. 3.12 - Local and Global Stereo Methods. In AL Bovik, editor, Handbook of Image and Video Processing (Second Edition), Communications, Networking and Multimedia, pages 297–308. Academic Press, Burlington, January 2005.

[21] P. Lottes, M. Hoeferlin, S. Sander, M. Muter, P. Schulze, and Lammers C. Stachniss. An effective classification system for separating sugar beets and weeds for precision farming applications. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 5157–5163, Stockholm, Sweden, May 2016. IEEE.

[22] Philipp Lottes, Raghav Khanna, Johannes Pfeifer, Roland Siegwart, and Cyrill Stachniss. UAV-based crop and weed classification for smart farming. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 3024–3031, Singapore, Singapore, May 2017. IEEE.

[23] F López-Granados. Weed detection for site-specific weed management: mapping and real-time approaches: Weed detection for site-specific weed management. Weed Research, 51(1):1–11, February 2011.

[24] Federico Magistri, Daniele Nardi, and Vito Trianni. Using prior information to improve crop/weed classification by mav swarms. In 2019 IMAV/11th INTERNATIONAL MICRO AIR VEHICLE COMPETITION AND CONFERENCE (IMAV), pages 67–75, 2019.

[25] Juul Maria. Civil drones in the European Union. European Parliament - Think Tank, page 8, 2015.

[26] K. McGuire, M. Coppola, C. de Wagter, and G. de Croon. Towards autonomous navigation of multiple pocket-drones in real-world environments. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 244–249, September 2017. ISSN: 2153-0866.

[27] A. Milioto, P. Lottes, and C. Stachniss. REAL-TIME BLOB-WISE SUGAR BEETS VS WEEDS CLASSIFICATION FOR MONITORING FIELDS USING CONVOLUTIONAL NEURAL NETWORKS. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, IV-2/W3:41–48, August 2017.

[28] Sharada P. Mohanty, David P. Hughes, and Marcel Salathé. Using Deep Learning for Image-Based Plant Disease Detection. Frontiers in Plant Science, 7, 2016. Publisher: Frontiers.

[29] Nvidia. Graphic card NVIDIA GeForce RTX 3090. `https://www.nvidia.com/fr-be/geforce/graphics-cards/30-series/rtx-3090/`. Accessed: 2021-04-15.

[30] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, 22(10):1345–1359, October 2010.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019.

[32] José Manuel Peña, Jorge Torres-Sánchez, Ana Isabel de Castro, Maggi Kelly, and Francisca López-Granados. Weed Mapping in Early-Season Maize Fields Using Object-Based Analysis of Unmanned Aerial Vehicle (UAV) Images. PLoS ONE, 8(10):e77151, October 2013.

[33] Phung and Rhee. A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. Applied Sciences, 9:4500, 10 2019.

[34] Francis J. Pierce and Peter Nowak. Aspects of Precision Agriculture. In Advances in Agronomy, volume 67, pages 1–85. Elsevier, 1999.

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497 [cs], January 2016. arXiv: 1506.01497.

[36] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 658–666, Long Beach, CA, USA, June 2019. IEEE.

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Lecture Notes in Computer Science, pages 234–241, Cham, 2015. Springer International Publishing.

[38] A Ruckelshausen, P Biber, M Dorna, H Gremmes, R Klose, A Linz, R Rahe, R Resch, M Thiel, D Trautz, and U Weiss. Bonirob: an autonomous field robot platform for individual plant phenotyping. Precision Agriculture, 9, 2009.

[39] Christian Scholz, Maik Kohlbrecher, Arno Ruckelshausen, Daniel Kinski, and Daniel Mentrup. Camera-based selective weed control application module ("Precision Spraying App") for the autonomous field robot platform BoniRob. In AgEng International Conference of Agricultural Engineering, page 8, 2014.

[40] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs], April 2015. arXiv: 1409.1556.

[41] Nima Tajbakhsh, Jae Y. Shin, Suryakanth R. Gurudu, R. Todd Hurst, Christopher B. Kendall, Michael B. Gotway, and Jianming Liang. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? IEEE Transactions on Medical Imaging, 35(5):1299–1312, May 2016.

[42] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective Search for Object Recognition. International Journal of Computer Vision, 104(2):154–171, September 2013.

[43] Guido Van Rossum and Fred L. Drake. Python 3 Reference Manual. CreateSpace, Scotts Valley, CA, 2009.

[44] Adam C. Watts, Vincent G. Ambrosia, and Everett A. Hinkley. Unmanned Aircraft Systems in Remote Sensing and Scientific Research: Classification and Considerations of Use. Remote Sensing, 4(6):1671–1692, June 2012.

[45] Ulrich Weiss and Peter Biber. Plant detection and mapping for agricultural robots using a 3D LIDAR sensor. Robotics and Autonomous Systems, 59(5):265–273, May 2011.

[46] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[47] Erol Şahin. Swarm Robotics: From Sources of Inspiration to Domains of Application. In Erol Şahin and William M. Spears, editors, Swarm Robotics, Lecture Notes in Computer Science, pages 10–20, Berlin, Heidelberg, 2005. Springer.